VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Informační systém pomocí orchestrace služeb

Information System Using Service Orchestration

2013 Stanislav Žabka

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Stanislav Žabka**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Informační systém pomocí orchestrace služeb
Information System Using Service Orchestration

Zásady pro vypracování:

Cílem práce bude nejprve nastudovat a popsat principy orchestrace služeb a následně jej otestovat v rámci implementace obchodního informačního systému.

1. Popište principy, standardy a jazyky používané u orchestrace služeb.
2. Porovnejte dostupné platformy a aplikační rámce, které se používají pro orchestraci služeb.
3. Vytvořte analýzu obchodního informačního systému.
4. Navrhněte a implementujte prototyp obchodního informačního systému s využitím orchestrace služeb na zvoleném aplikačním rámci.

Seznam doporučené odborné literatury:

JURIC, Matjaz B, Benny MATHEW a Poornachandra SARANG. *Business process execution language for web services: an architect and developer's guide to orchestrating web services using BPEL4WS*. 2nd ed. Birmingham: Packt Publishing, c2006, x, 353 p. From technologies to solutions. ISBN 19-048-1181-7.
SADIQ, Waqar, Felix RACCA a Poornachandra SARANG. *Business services orchestration: the hypertier of information technology*. 2nd ed. New York: Cambridge University Press, 2003, xx, 380 p. From technologies to solutions. ISBN 05-218-1981-4.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Lumír Návrat**

Datum zadání: 18.11.2011
Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Súhlasím so zverejnením tejto diplomovej práce podľa požiadavkov čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava*.

Diplomová práca je bez obmedzenia prístupu a jej zverejnením neporušujem žiadne autorské práva

V Ostrave 1. mája 2013 .....

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave 1. mája 2013 .....

Touto formou by som sa chcel poďakovať Ing. Lumírovi Návratovi za vedenie, poskytnutie odbornej pomoci a potrebných materiálov.

Abstrakt

Práca sa zaoberá problematikou orchestrácie webových služieb a jej využitia v informačnom systéme. Prvá časť je zameraná na zoznámenie čítateľa s teóriou webových služieb, servisne orientovanej architektúry a samotnej orchestrácie. Následuje prehľad jazykov, prístupov, dostupných platforiem a rámcov. Druhá časť predstavuje návrh a implementáciu prototypového informačného systému, ktorým ukážem výhody, možnosti i problémy spojené so zvolenou technológiou. Komponenty sú napísané v jazyku Java a pre orchestráciu som využil WSBPEL 2.0. V práci nájdete príklady použitia rôznych štruktúr, volaní i operácii s premennými. Snažil som sa ukázať implementácie reálnych procesov v prostredí obchodných informačných systémov.

Kľúčové slová: biznis, BPEL, databáza, ESB, informačný systém, integrácia, Java, návrh, orchestrácia, proces, servisne orientovaná architektúra, volanie, webová služba, WSDL

Abstract

The work deals with problematic of web service orchestration and its usage in information system. First part is aimed to familiarize reader with theory of web services, service oriented architecture and orchestration itself. Next is language, approach, platform and framework overview. Second part presents design and implementation of prototype system, which shows advantages, possibilities and problems connected with selected engine. Components are written in Java and I used WSBPEL 2.0 for orchestration. You will find examples of structures, calls and also operations with variables. I tried to show real process implementation in business systems environment.

Keywords: business, BPEL, call, database, design, ESB, information system, integration, Java, orchestration, process, service oriented architecture, web service, WSDL

Zoznam použitých skratiek a symbolov

BPEL	–	Business Process Execution Language
BPM	–	Business process management
ESB	–	Enterprise service bus
SOA	–	Servisne orientovaná architektúra
SOAP	–	Simple Object Access Protocol
UDDI	–	Universal Description, Discovery, and Integration
WS	–	Webová služba
WSDL	–	Web Services Description Language
WSO	–	Web Services Orchestration
XML	–	eXtensible Markup Language

Obsah

Zoznam tabuliek

Zoznam obrázkov

Zoznam výpisov zdrojového kódu

1 Úvod

Dnešná doba veľkých podnikov, ktoré sú nútené neustále svoje biznis procesy prispôsobovať trhu a ostať konkurencie schopné, vnáša potrebu podobne flexibilných riešení i v oblasti informačných systémov. Väčšinou každá spoločnosť používa niekoľko programov, predstavujúce najlepšiu volbu pre jednotlivé oddelenia a ich štýl práce. Dôvody sú historické, finančné alebo štrukturálne. Súčasnosť so sebou prináša nutnosť integrácie týchto systémov, či už z dôvodu zdielania dát, závislosti medzi nimi, evidenicie, štatistík a iných výstupov pre ekonómov, proces manažerov i ďalších vedúcich pracovníkov firiem. Moja práca je zameraná práve na prepojenie jednotlivých častí podnikových systémov alebo rôznych softvérov. K temuto účelu som zvolil orchestráciu webových služieb.

Teoretická časť obsahuje zhrnutie problematiky služieb, ktoré do procesu vstupujú ako poskytovatelia alebo konzumenti dát. Ďalej je tam popísaná servisne orientovaná architektúra, základy, postupy a jazyky používané pri orchestrácii. Obsahuje i prehľad dostupných platforiem a rámcov od veľých firiem ako Oracle, JBoss a Microsoft.

Praktická časť je zameraná na návrh a implementáciu prototypového informačného systéme zvolenou technológiou, pričom dôraz bol kladený na nezávislosť a konfigurovatelnosť, odpovedajúcu potrebám procesov danej firmy. Cieľom je ukázať možnosti, výhody, postupy a problemý spojené z použitim orchestrácie webových služieb.

2 Webové služby

Webové služby (web services) sú služby poskytované v prostredí Internetu. Jedná sa o technológiu, ktorá umožňuje sprostredkovanie funkcionality na aplikačnej alebo biznis úrovni. Dôležité je použitie štandardných rozhraní a jednoduchého spôsobu prenosu pomocou internetových protokolov. Základnými vlastnosťami WS sú: modularita, prístupnosť, dobrý popis, implementačná nezávislosť a znovupoužitelnosť. V podstate ide o komunikáciu medzi dvoma počítačmi, pri ktorej má jeden funkciu poskytovateľa a druhý je klient. Poskytovateľ služby poskytuje dáta špecifikovaným spôsobom na sieti. Na druhej strane si klient zistí adresu služby (vyhľadá v registri alebo má adresu priamo od poskytovateľa), stiahne si popis služby a následne ju môže používať.

Každá služba sa skladá z troch komponent:

- Rozhranie
  - Toto definuje, ako môže byť vystavená a sprístupnená klientom.
  - Rozhranie nie je limitované webovými službami and môže byť reprezentované ľubovolným vzdialeným protokolom správ.
- Kontrakt (dohoda, spolupráca)
  - Definuje, čo služba očakáva počas komunikácie s klientom. Štruktúry správ, pravidlá a bezpečnostné mechanizmy sú takisto súčasťou kontraktu..
  - Kontrakt definuje "legálnu" dohodu o tom, ako bude služba a klient spolupracovať.
- Implementácia
  - Je to vlastný kód služby.

Technológia WS je založená na **XML/SOAP** pomocou protokolov internetu. WS protokol pozostáva so sady protokolov, ktoré sa využívajú v SOA (Service Oriented Architecture). Jedným zo spôsobov predávania správ je XML a XML Schéma, ktoré sa používajú k tvorbe dokumentov popisujúcich spôsob komunikácie medzi službami. Základná funkcionalita SOA je riešená správami, popisnou vrstvou a viditeľnou časťou procesnej vrstvy. Rozhranie WS je definované WSDL (Web Service Definition Language), v adresári WS UDDI (Universal Description, Discovery and Integration) nájdeme relevantné služby a zavoláme ich správami definovanými v SOAP (Simple Object Access protokol).

2.1 Web Services Description Language

Keďže komunikačné protokoly a správy majú štandardizovaný formát, bolo potrebné zaviesť určité pravidlá a štruktúru v komunikácií medzi službami. WSDL adresy majú definovanú gramatiku pre popisovanie sieťovej komunikácie ako kolekcie komunikujúcich koncových bodov schopných výmeny správ. WSDL definícia služieb poskytuje

dokumentáciu pre distribuované systémy a slúži ako návod pre automatizáciu detailov zahrnutých v komunikácií aplikácií.

WSDL dokument definuje služby ako kolekciu sieťových koncových bodov alebo portov. Definícia sa delí na:

- **Abstraktná** – udržuje integritu popisu služby. Definícia koncových bodov a správ je oddelená od ich konkrétneho sieťového nasadenia alebo naviazania na dátový formát. To umožňuje znovupoužitie abstraktných definícií. Správy sú abstraktným popisom dát ktoré si budú medzi sebou posielať a typy portu sú abstraktnou kolekciou operácií.
- **Konkrétna** – slúži k naviazaniu logiky z abstraktnej definície na konkrétnu implementáciu a komunikácie na konkrétny protokol. Port je definovaný asociáciou k sieťovej adrese so znovupoužiteľnou väzbou a kolekcia portov definuje službu.

WSDL dokument používa následujúce elementy k definícií sieťových služieb:

- Typy – kontajner pre definovanie dátových typov
- Správy – abstraktná definícia typu dát, ktoré budú posielané
- Operácie – abstraktný popis akcie podporovanej danou službou
- Väzba – abstraktná sada operácií podporovaných koncovými bodmi.
- Typ portu – konkrétny protokol a dátový formát pre port
- Port – koncový bod definovaný ako kombinácia väzby a sieťovej adresy
- Služba – kolekcia príbuzných koncových bodov

WSDL nevyužíva nový jazyk pre definovanie typov, ale používa XML Schemas definition ako základný typový systém. Popis viacerých formátov pomocou WSDL je možný prostredníctvom rozšírení.

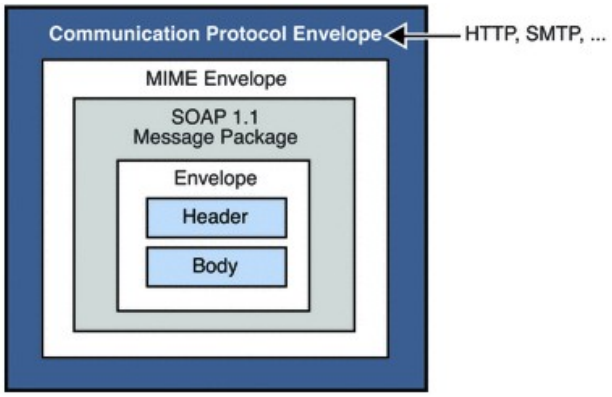
2.2 Simple Object Access Protocol

SOAP vo verzii 1.2 poskytuje definíciu informácií založených na XML, ktoré môžu byť použité pre výmenu štruktúrnych a typových informácií medzi uzlami v decentralizovanom, distribuovanom prostredí. V našom prípade sa jedná o definíciu štruktúry a typu správ medzi službami. SOAP správa je formálne špecifikovaná ako XML Information Set, ktorý posyktuje popis jej obsahu.

Vlastnosti SOAP:

- Je vyvíjaný so zameraním na jednoduchosť a ľahkú rozšíriteľnosť
- Nezávislý na transportných protokoloch. HTTP je len jedným z podporovaných podporných protokolov. SOAP správy je možné posielať i emailom.
- Bezstavový protokol
- SOAP je nezávislý na operačnom systéme

Štruktúra SOAP správy:



Obr. 1: SOAP správa [12]

Hlavička obsahuje dva bloky, každý definovaný vo vlastnom XML mennom priestore a obsahuje dáta týkajúce sa obsahu správy, ktorý je obsiahnutý v jej tele.

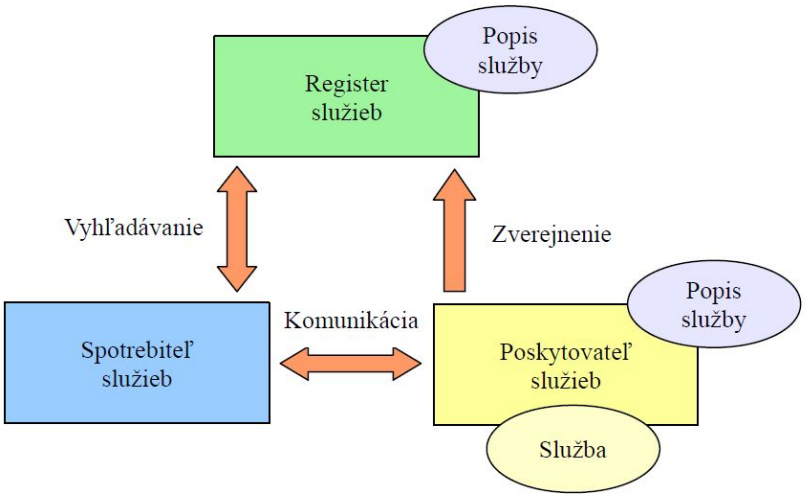
2.3 Universal Description, Discovery, and Integration

UDDI je biznis register založený na XML pre vyhľadávanie na Internete. Popisuje mechanizmus adresára služieb. V podstate ide o webovú službu, ktorá poskytuje informácie o ďalších službach na internete. Je možne vyhľadávať podľa rôznych kritérií (názov, oblasť záujmu, . . .).

Keďže UDDI register bol plne verejný, záznamy mohol pridávať prakticky ktokoľvek, čo viedlo k tomu, že väčšina záznamov nebola korektná a relevantná. Preto v roku 2006 IBM, Microsoft a SAP (hlavný poskytovatelia) túto službu verejného registru vypli. Špecifikácia UDDI nebola nikdy štandardizovaná a dnes sa používa maximálne na vnútropodnikovej úrovni.

3 Servisne orientovaná architektúra

Základom sú webové služby. V SOA (servisne orientovanej architektúre) nie je požadovaná funkcionalita poskytovaná ako jedna mohutná aplikácia. Skladá sa z jednotlivých služieb, ktoré zabezpečujú jednotlivé funkcie. Tie môžu záujemcovia (klienti) nájsť v registri služieb.



Obr. 2: Princíp SOA

Tento koncept je okrem použitia v sofvérovej architektúre aplikovateľný i na biznis úrovni. Je postavený na princípe voľne viazaných, opakovane použiteľných, definovaných a na štandardoch založených služieb, ktoré sú dostupné a využiteľné nezávislými spotrebiteľmi. SOA umožňuje organizáciám vhodne prepojiť obchodné a IT služby a zabezpečiť tak v prostredí neustálych zmien schopnosť riadenia, stability, predpovedateľnosti a bezpečnosti.

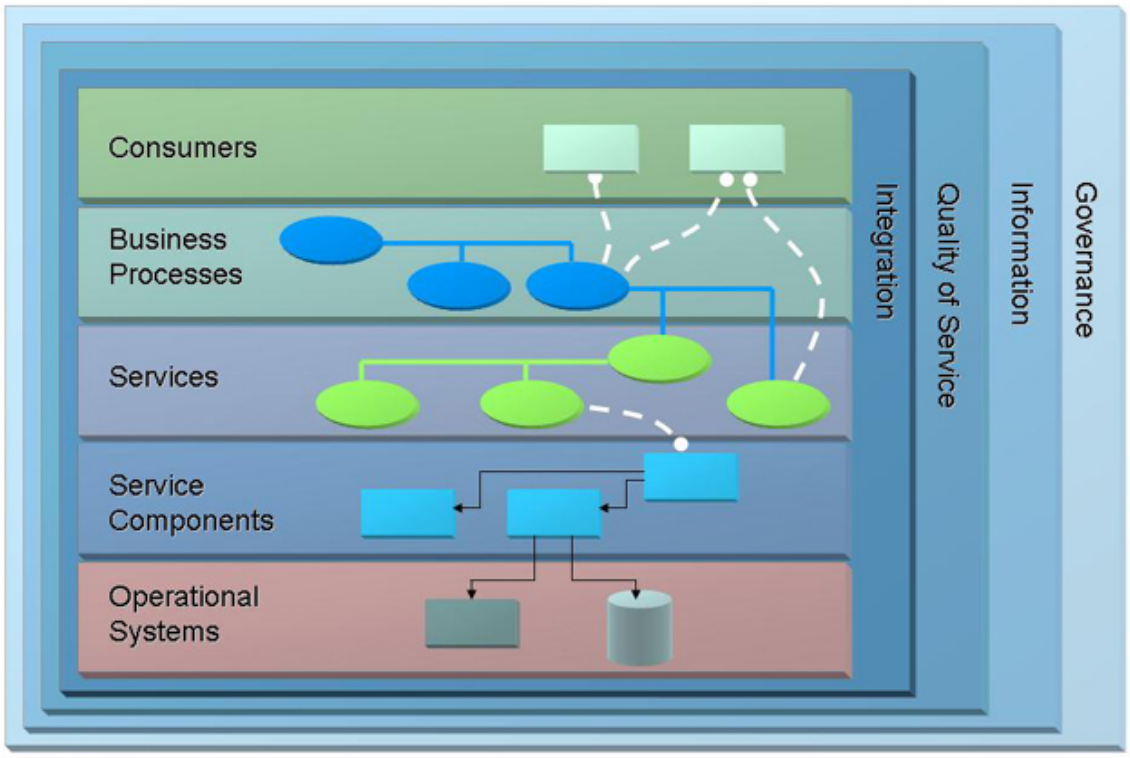
3.1 Služba v SOA

Služba je v SOA základný stavebný prvok. Z pohľadu servisne orientovanej architektúry je služba vhodne zvolená, granulovaná funkcionalita buď existujúceho IS, ktorú zverejňuje, poskytuje okolitému svetu, alebo vhodne vytvorená nová funkcionalita vytvorená z iných služieb. SOA umožňuje ich opakovane skladanie a vytváranie nových, komplexnejších služieb.

Kompozitná služba – je zložená s viacerých služieb, takisto môže byť poskytovaná. Zabezpečuje mechanizmy pre monitorovanie, kontrolu zhody a koordinovanie rozdielnych podslužieb.

Servisný manažment – poskytuje služby ako zabezpečenie, certifikovanie a hodnotenie služieb.

Jeden z možných pohľadov na SOA je čiastočne vrstvená architektúra.



Obr. 3: SOA architektúra [19]

Najnižšiu vrstvu predstavujú operačné systémy, ktoré obsahuje zdroje pre beh celého informačného systému a dáta (najčastejšie vo forme databázy), s ktorými IS pracuje. Vyššie nájdeme komponenty. Sú základné stavebné zložky služieb, zaobalujú funkcionalitu poskytovanú službami a zaisťujú ich požadované kvality (QoS). Komponenty si môžeme predstavit'ako čierne skrinky a k funkciám pristupujeme len cez ich rozhranie. Nad komponentami sa nachádza vrstva služieb. Služba zostavuje komponenty a preposiela im požiadavky, ktoré sú od nej žiadané. Má tiež svoje rozhranie a na jeho základe vzniká popis používaný pri komunikácii, vyhľadávaní a riadení. Vrstva biznis procesov predstavuje množinu biznis cieľov, ktoré môžeme pomocou služieb dosiahnuť. **Biznis proces** je v SOA kontexte reprezentovaný ako sekvencia vykonávania niekoľkých služieb. Najvyššie je vrstva Zákazníkov. Tí posielajú požiadavky a čakajú na odpoveď (výsledok, akcia systému,...).

Celá architektúra obsahuje ešte štyri riadiace vrstvy. Zabezpečujú integrácia komponent a služieb do väčších celkov. Riadia a monitorujú SOA aplikácie (QoS), poskytujú informácie (rozhrania komponent a služieb, . . .) a spravujú komunikáciu medzi vrstvami.

Prínosy SOA

- Opakovateľné použitie služieb = zníženie nákladov
- Zníženie nákladov na vývoj a údržbu
- Možnosť opätovného použitia služieb
- Eliminácia duplicít vo vývoji
- Lepšia kontrola a transparentnosť
- Zvýšenie agility procesov = pružnejšia organizácia
- Schopnosť rýchlo realizovať navrhnuté obchodné procesy
- Variantnosť, flexibilita, rýchle zavádzanie zmien, automatizácia
- Merateľnosť procesov = riadenie procesov
- Nastavenie a monitorovanie merateľných ukazovateľov procesov
- Optimalizácia procesov z hľadiska merateľných ukazovateľov

3.2 Spolupráca medzi službami

Služby ako základný stavebný prvok SOA architektúry vzájomne komunikujú pomocou zasielania správ. Táto spolupráca môže byť rozdelená na štyri typy [2]:

- **Kooperácia** – služba používa prostriedky inej služby, na realizáciu svojej funkcionality
- **Agregácia** – zostavenie novej služby, spojením viacerých služieb
- **Choreografia** – spolupráca medzi službami za účelom vykonania biznis procesu.
- **Orchestrácia** – nejedná sa tak o spoluprácu medzi službami, ako o riadenie a koordináciu aktivít nadradeným procesom.

V práci sa ďalej zameriam práve na orchestrácia služieb.

4 Orchestrácia webových služieb

Vyššie spomenuté technológie ako WSDL, SOAP, UDDI pracujú s webovými službami a poskytujú nám prostriedky pre ich jednotlivý popis, lokalizáciu a spúšťanie. I keď môže ponúkať veľa metód, každý WSDL súbor popisuje doslova atomické (na nízkej úrovni) funkcie. Čo nám však tieto základné technológie neposkytujú, sú dôležité detaily, ktoré popisujú správanie služby ako súčasti väčšieho a komplexnejšieho celku. Obyčajným statickým spájaním funkcionality nie sme schopní využiť ich potenciál, nie ešte potenciál servisne orientovanej architektúry (SOA). Preto je potrebné začať služby reťaziť dynamicky, tzv. spájať ich podľa aktuálnych potrieb, možností užívateľa. Keď sa jedná o spoluprácu, ktorá je kolekciou aktivít (metód, služieb) navrhnutých tak, aby úspešne plnila daný business ciel, jedna sa o tzv. business proces.

Orchestrácia webových služieb je centrálny proces (môže sa jednať o ďalšiu webovou službu) preberajúci kontrolu nad službami, ktoré sú do procesu zapojené, a koordinuje spúšťanie jednotlivých operácií. Zúčastnené služby nevedia, a ani nemusia vedieť, že sú účastníkmi nejakého vyššieho procesu [Šafář]. Podľa Peltza [Peltz] orchestrácia zahŕňa poradie vykonávania interakcií webových služieb, popisuje tok vykonateľného procesu a môže zahŕňať ako interné, tak externé webové služby. Proces je vždy riadený jednou stranou. Interakcie nastávajú na úrovni správ. Zahŕňajú biznis logiku, poradie vykonávania úloh a môžu pokryť aplikácie a organizovanie k definovaniu dlhotrvajúceho, transakčného a viacstupňového procesného modelu.

V jednoduchosti povedané, ide o automatizovaný spôsob ako kombinovať služby a vytvoriť tak novú funkcionalitu. V závere nám vznikne nová kompozitná služba, ktorá poskytuje odlišnú biznis logiky, vystupujúcu samostatne s vlastným rozhraním.

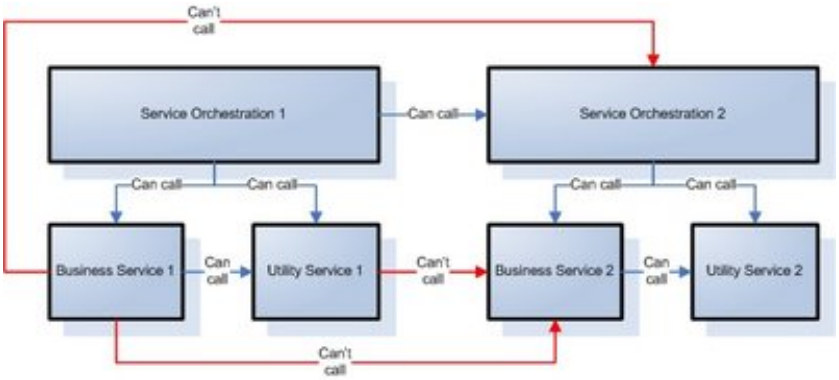
Orchestrácia predstavuje technologicky nezávislý koncept. Môže byť dosiahnutý pomocou popisného jazyka, ako je **BPEL**, zabudovanými nástrojmi vrámci špecifickej platformy (ESB poskytuje vlastné mechanizmy), alebo manuálnym naprogramovaním. V závislosti od potrieb, situácie a dostupných technológií, sa spôsob orchestrácie líši. Snahou je vytvoriť ju čo najrýchlejšie a pritom zachovať flexibilitu, udržatelnosť a škálovatelnosť.

Použitie platformy so vstavanými orchestračnými schopnosťami je prvá možnosť. Snažte sa vyvarovať implementácie WSO programovaním. Zvolte platformu alebo mechanizmus, ktorý jednoducho zvládne logiku tokov dát, agregáciu výsledkov, transformáciu správ a aplikovanie biznis pravidiel. Mala by v sebe obsahovať väčšinu funkcionality, potrebnú pri tvorbe novej orchestrácie služieb. Uistite sa, že kompozitná služba zodpovedá definícií služby (má všetky vlastnosti služby). Typicky budete volať niekoľko služieb, agregovať ich výsledky alebo reťaziť ich volanie pomocou nejakej logiky, transformovať výsledky pre potreby klientov a vracať ich. Čím menej práce musíte urobiť a čím viac toho zvládne zvolená platforma, tým efektívnejšia bude vaša orchestrácia. Cieľom je, aby ste strávili menej času správou, a tým pádom bude jednoduchšie urobiť požadované zmeny a nemusíte budovať všetky potrebné mechanizmy od základu.

Niekto môže tvrdiť že programovací jazyk poskytuje väčšiu flexibilitu pri implementácií. Na jednej strane je to pravda, ale je to neúmerné množstvo práce a nízka efektívnosť. Žiadny programovací jazyk úplne neintegruje všetky mechanizmy, ktoré potrebujete na vytvorenie orchestrácie, zvlášť už nie vizuálne. A navyše, zakaždým keď bude nutna

zmena, bez ohľadu na to o aký rozsah ide, musíme napísať a otestovať nový kód. Pri použití niektorej známej platformy bude stupeň náročnosti oveľa menší.

Keď vytvárame orchestráciu, je dôležité zabezpečiť správne vzťahy medzi kompozitnými a atomickými (základnými) službami. Na obrázku nižšie môžete vidieť, ako by mali služby spolu komunikovať.



Obr. 4: Komunikácia služieb [1]

Pravidlá orchestrácie [1]:

1. *Atomické biznis* služby by sa nemali volať navzájom. Na tento účel slúži orchestrácia. Pokiaľ sa služby môžu volať jedna druhú, nezískame tým výhody, ktoré prináša daná platforma. Navyše sa tým služby stávajú previazanejšie a zmenšujú sa možnosti ich použitia a údržby.

2. *Biznis služby* môžu volať utilitné služby. I keď sa snažíme vyhnúť vzájomnému prepájaniu základných služieb, niekedy potrebujeme generické, nízko úrovňové funkcie služieb vystaviť pomocou utilitných služieb. Niekdy by bolo nemožné použiť orchestráčnú platformu na umožnenie biznis službám využiť výhody funkcionality ako napríklad logovanie, získanie alebo uloženie konfiguračných informácií a autorizácie.

3. *Utilitné služby* nemôžu volať biznis služby. Utilitné služby by nemali byť previazané so žiadnymi biznis procesmi a schopnosťami. Preto by utilitné služby volajúce biznis služby toto pravidlo porušili.

4. *Biznis služby* nemôžu volať kompozitné služby. Dôvod je ten istý ako pri volaní služieb navzájom pretože kompozitná služba je tiež biznis služba.

5. *Kompozitné služby* môžu volať iné kompozitné služby. Tieto služby sa môžu podieľať na orchestrácií. Mali by sa chovať ako regulárne atomické služby.

Ako už vyplýva z vyššie uvedených pravidiel, sú určité rozdiely medzi atomickými a kompozitnými službami, ale obidvoje sú biznis služby. Existujú dva spôsoby pohľadu pri ich porovnávaní – logický a fyzický. Z logického hľadiska nie je medzi nimi žiadny rozdiel.

Ponúkajú určitú jedinečnú biznis funkcionalitu a dodržujú definíciu služieb ako takých. Z fyzického pohľadu sa atomická biznis služba a kompozitná služba líšia. Atomická je súčasť kompozitnej, má na starosti vnútornú biznis logiku a priamo pracuje s dátovými zdrojmi. I keď výsledok práce kompozitnej služby závisí od ostatných biznis služieb, nemal by byť rozdiel vo volaní atomickej alebo kompozitnej služby.

4.1 Výhody orchestrácie

WSO umožňuje biznisu zamerať sa z dátových na procesné aplikácie. Dôvody pre orchestráciu a orchestračné nástroje v priemysle a službách sú:

- *Komplexnosť riešenia a jeho použitia* umožňuje pokryť široký okruh biznis domén.
- *Lepšie usporiadanie biznisu* na základe procesov. Biznis procesy poskytujú základ pre dizajn korektne navrhnutej SOA.
- *Zlepšenie flexibility* pomocou vymedzenia a zaobalenia procesnej logiky. Jednotlivé časti systému sú logicky oddelené a ich znovupoužitie a prepojenie s inými službami či systémami sa stáva jednoduchšie. Je tým zabezpečená rýchlejšia odpoveď na zmeny v biznis prostredí.
- *Zvýšenie možnosti na špecializáciu* pomocou outsorcingu vedľajších služieb. Špecializácia umožňuje spoločnostiam vybrať si svojich dodávateľov a biznis partnerov, čo sa prejaví na nižších nákladoch a vyššej kvalite.
- *Nižšie náklady na vlastníctvo* pomocou kompozície služieb. Dostupnosť množiny biznis služieb znižuje náklady na tvorbu nových procesov.
- *Zdokonalené riadenie* poskytnutím Key Performance Indicators (KPI). Meranie výkonnostných parametrov je kritická časť pre úspešné porozumenie a zlepšenie biznis procesov, a to ako sú ovplyvnené inými aktivitami.

4.2 Problémy spojené s orchestráciou

Orchestrácia je komplexný, robustný a inovatívny spôsob návrhu a tvorby informačných systémov. Na druhú stranu, tak ako aj pri iných riešeniach, existujú určité komplikácie a nie vždy je vhodné použiť tento prístup. Zoznam možných problémov:

- *Komplexnosť* daného riešenia spôsobuje, že na jeho realizáciu je nutné veľké množstvo softvéru, doplnkov i výkonného hardvéru, ktorý zvládne generovanú záťaž. Môžeme spomenúť vývojové prostredia s množstvom prídavných knižníc, rozšírení a podporných pluginov. A na druhú stranu sú tu aplikačne a databázové servery doplnené o rôzne registre a špeciálne organizačne a riadiace moduly.
- *Znalosti a skúsenosti vývojárov* v danej problematike sú vysoko cenené, z dôvodu robustnosti a množstva technických aspektov.
- *Rozsiahlosť* orchestrácie môže vnášať do projektu zložitosť a neprehľadnosť. Preto je dôležité bezchybné projektové riadenie.
- *Časová náročnosť* na analýzu, návrh a vývoj sú zjavné z predchádzajúcich bodov.
- *Cena* je jeden z hlavných dôvodov, ktoré vedú k zamietnutiu orchestrácie ako spôsobu tvorby IS zákazníkom. Tá ale zodpovedá prostriedkom a úsiliu, ktoré musia vývojári vynaložiť. Na druhú stranu, je ale cenové riešenie prijateľnejšie pre ďalší rozvoj systému.
- *Použiteľnosť* je určite výhodnejšia pri veľkých projektoch, kde sa ráta s rozširovaním, zmenami a dlhodobým nasadením. Pri malých systémoch môže byť orchestrácia finančne i časovo nevýhodná.

4.3 Druhy orchestrov

Základné delenie orchestrov je na 3 druhy [16]. Patria sem statické orchestry, jednoduché dynamické a pokročilé dynamické orchestry. Statické orchestry sa v súčastnosti bežne používajú a dynamický prístup k orchestrácií služieb bol prvý krát v rámci ČR popísaný v roku 2009 v metodike *Orchestrace geowebových služeb* riešenia grantu GAČR 205/07/0797.

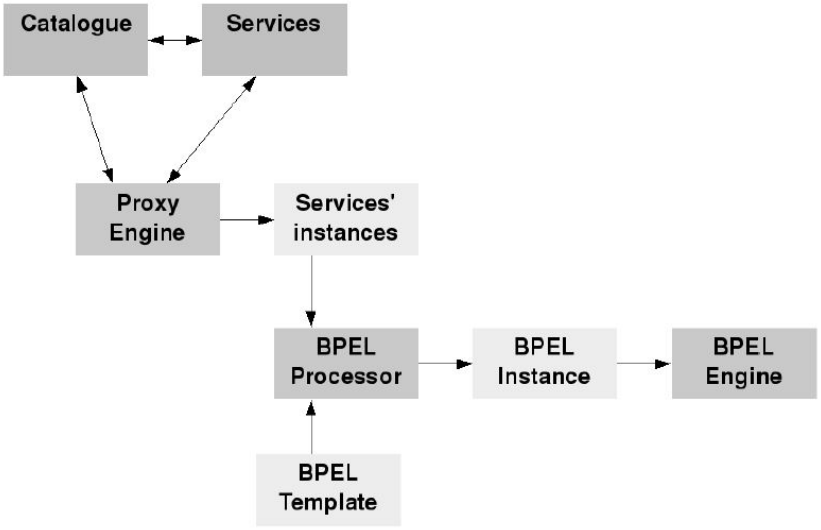
Statická orchestrácia

Pri statických orchestroch je sada inštancií služieb zapojených do orchestru presne definovaná a nemenná. Vždy sa spúšťa rovnaká inštancia služby, bez ohľadu na efektívnosť alebo konkrétne potreby rôznych užívateľov.

Dynamická orchestrácia

Dynamická orchestrácia je založená na princípe, kde sa aktuálna inštancia služieb volí až v priebehu spúšťania orchestru. Vďaka tomu je možné optimalizovať dátové toky alebo vyberať služby a ich metódy, ktoré viac vyhovujú aktuálnemu užívateľovi. Biznis proces je definovaný pomocou BPEL Templatu, čo je bežný BPEL súbor, ktorý obsahuje parametre pre vyľadenie inštancií služieb, ktoré sú do procesu zapojené. BPEL Template je spracovaný BPEL Procesorom, ten využíva služieb Proxy Engine, ktorý vyhľadáva v Katalógu požadované inštancie služieb, prípadne sama vytvára inštancie služieb. Zostavenie inštancie BPEL procesu má na starosti BPEL Procesor na základe BPEL Template a adries inštancií služieb, ktoré získa z Proxy. Následne je proces spustený pomocou bežného BPEL Enginu.

Architektúra dynamickej orchestrácie je popísaná na následujúcom obrázku.



Obr. 5: Dynamická orchestrácia [16]

5 Jazyky a prístupy

Orchestrácia webových služieb nie je primitívny spôsob ich organizovania a preto vyžaduje určité pravidlá a postupy pri jej návrhu a vývoji. Vhodne zvoleným jazykom môžeme popísať závislosti a interakcie medzi jednotlivými časťami systému. Prístupov k orchestrácií je niekoľko a výber postupu a techniky je podmienený systémom, ktorý vytvárame, aplikačnou doménou i programátorovým úsudkom a skúsenostami.

5.1 Business Process Execution Language

Na základe spolupráce firiem IBM, BEA Systems a Microsoft vznikol štandardizovaný jazyk Business Process Execution Language (BPEL). BPEL je v súčastnosti podporovaný širokým spektrom komerčných subjektov. Môžeme pomocou neho definovať jednoduché i veľmi zložité procesy. BPEL je výrazne podobný tradičným programovacím jazykom, obsahuje cykly, vetvenie, premenné, priradenie atď. Tieto konštrukcie nám dovoľujú namodelovať prakticky ľubovolný proces. A však najdôležitejšou vlastnosti sú spojené s volaním webových služieb. BPEL sa v poslednej dobe stal významným štandardom, podporujúci využitie SOA z IT úrovne na biznis úroveň. Umožňuje organizáciám spustenie a automatizovanie ich biznis procesov, prostredníctvom orchestrácie služieb vnútri i mimo danej organizácie. Tento jazyk umožňuje špecifikovať ako spustiteľné, tak abstraktné procesy.

- Abstraktný proces je čiastočne špecifikovaný proces, u ktorého sa nepredpokladá, že bude niekdy spustený. To nám umožňuje popísať úlohy, ktoré používame vo viacerých prípadoch použitia.
- Spustiteľný proces výlučne spolieha na zdroje webových služieb a na XML dáta.

Pomocou BPEL definujeme model a prostriedky pre popis chovania procesu, založeného na spolupráci medzi daným procesom a jeho partnermi. Spolupráca medzi všetkými partnermi je sprostredkovaná rozhraniami webových služieb a štruktúra spojenia na tejto úrovni je zapuzdrená do takzvaného "partnerLink". BPEL proces definuje ako všetky tieto spolupráce koordinovať, aby bolo dosiahnuté daného biznis cieľa, a tiež obsahuje všetkú potrebnú logiku. Obsahuje i mechanizmy na prácu s výnimkami a chybami.

Pri svojej činnosti využíva niekoľko **XML** špecifikácií: WSDL 1.1, XML Schema 1.0, XPath 1.0 a XSLT 1.0. WSDL správy a XML Schema poskytujú dátový model BPEL procesov. XPath spolu s XSLT ho rozširujú o manipuláciu s dátami. Všetky externé zdroje a partneri sú reprezentované ako WSDL služby.

Tie môžeme volať dvojím spôsobom, synchrónne a asynchrónne. Môžeme spúšťať operácie ako sekvenčne, tak paralelne. Po asynchrónnom volaní máme možnosť čakať na tzv. callback (spätné volanie). BPEL taktiež disponuje bohatou výbavou v oblasti obsluhy chýb, čo je veľmi dôležité pri vytváraní robustných biznis procesov, a poskytuje podporu pre ich dlhé trvanie.

BPEL teda umožňuje:

- popisovať biznis procesy pomocou skladania služieb,
- skladať väčšie procesy zo služieb a už vytvorených procesov,
- pracovať so synchrónnymi a asynchrónnymi operáciami a prijímať tzv. callbacks,
- spúšťať služby sekvenčne alebo paralelne,
- kompenzovať služby v prípade chyby,
- presmerovať prichádzajúcu správu patričnému procesu, pracovať s udalosťami,
- spúšťať aktivity v určitom poradí či za určitý čas,
- štrukturovať biznis procesy.

Aktuálna **verzia** je BPEL 2. Priebeh vývoja môžete vidieť tu:

1. 2002 - BPEL4WS 1.0 - IBM, Microsoft, BEA
2. 2003 - BPEL4WS 1.1 - OASIS
3. 2004 - WS-CDL - W3C (Kandidát)
4. 2007 - WSBPEL 2.0 - OASIS

5.1.1 Štruktúra definície procesu v BPEL

Dokument pre definíciu procesu v BPEL4WS má nasledujúcu štruktúru:

```
<process>
  <partnerLinks>
    ...
  </partnerLink >
  <variables>
    ...
  </variables>
  <correlationSets>
    ...
  </correlationSets>
  <faultHandlers>
    ...
  </faultHandlers >
  <compensationHandler>
    ...
  </compensationHandler>
  <eventHandlers>
    ...
  </eventHandlers>
  <sequence>
    <receive ...>
```

```
        <invoke ...>
        <reply ...>
         ...
    </sequence>
     ...
</process>
```

Výpis 1: Štruktúra BPEL dokumentu

Koreňový element `process` a jeho atribúty slúžia k pomenovaniu dokumentu a definícií menných priestorov použitých v dokumente. V tele sa ďalej nachádzajú tieto elementy:

- `partnerLinks` - určený pre výpis partnerských služieb, ktoré sa podieľajú na danom biznis procese. Partnerská služba môže vystupovať ako klient vzhľadom k inej službe biznis procesu (vyvoláva službu), alebo je naopak volaná biznis procesom. Túto vlastnosť určujú atribúty `myRole` (partnerská služba volá službu biznis procesu) resp. `partnerRole` (proces volá danou službu). Aby bolo možne túto konštrukciu použiť musíme upraviť popis služby vo WSDL dokumente. Zmenu vykonáme vložením oddielu `partnerLinkType` do kořenového oddielu `description`. Ten obsahuje zoznam rolí, pod ktorými služby v biznis procese vystupujú.

- `variables` – definujú sa tu premenné pre ukladanie informácií o/od služieb. Typ informácií je predom špecifikovaný. BPEL umožňuje deklarovať promenné tromi spôsobmi: ako typ WSDL správy, ako typ XML Schema a ako XML Schema element.

- `correlationSets` – zabezpečuje doručovanie prichádzajúcich správ odpovedajúcim inštanciám procesu. Dôvodom je, že pred samotným spustením daného procesu dochádza k vytvoreniu jeho inštancie.

- `faultHandlers` – má na starosť správu výnimiek, ktoré nastanú pri behu procesu. Chyby môžu byť vyvolané explicitne (pomocou `<throw>`) alebo implicitne (napr.: chyba pri volaní služby). Zachytávanie je riešene pomocou elementu `<catch>`.

- `compensationHandler` – poskytuje možnosť zotavenia a vrátenie procesu do stavu, kde môže po chybe pokračovať v behu.

- `eventHandlers` – slúži k zachytávaniu udalostí. V BPEL nájdeme dva typy: prichádzajúca správa (korešpondujúca s WSDL operáciami) a alarmy (aktivované po užívateľom definovanom čase). V každom elemente musí byť aspoň jeden z týchto typov.

- `sequence` – obsahuje zapísanú postupnosť aktivít, ktoré majú byť vykonané.

Zoznam základných aktivít:

invoke	identifikuje operácie partnerskej služby, ktorá má byť vyvolaná
receive	Umožňuje procesu čakať na odpoveď od partnerskej služby.
reply	Odoslanie odpovedi procesu na dotaz
assign	Používa sa pre prácu s premennými (priradenie hodnôt)

Tabuľka 1: Tabuľka aktivít

5.2 BPM engine

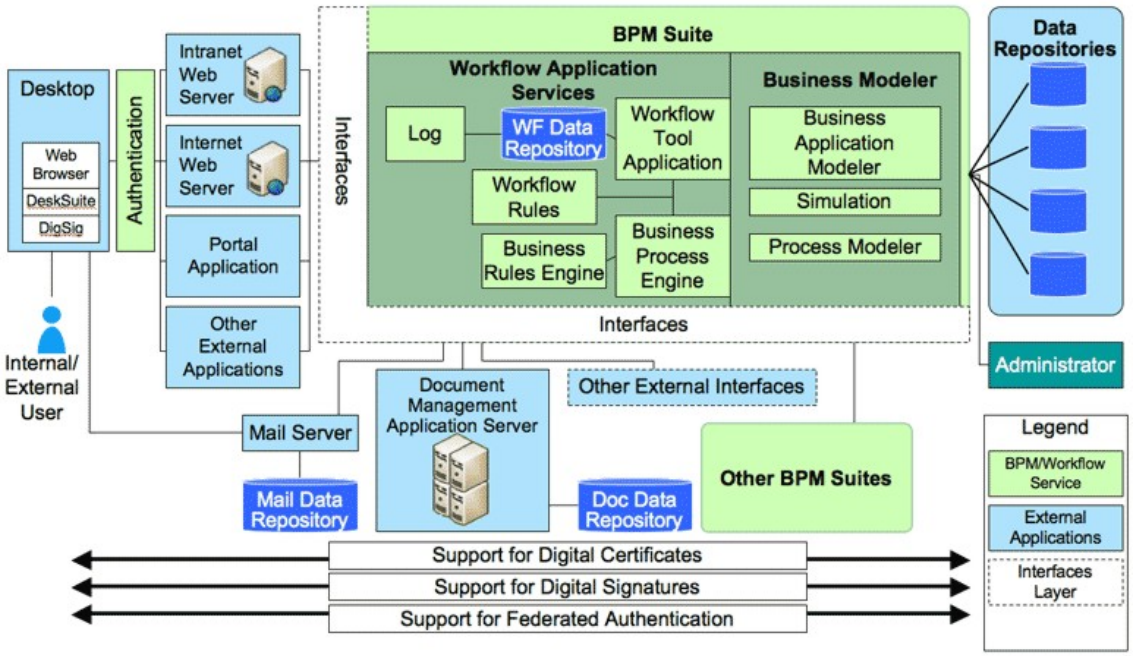
SOA v spojení s Business Process Management (BPM) umožňuje využitie hlavných procesných aplikácií, ktoré poskytujú najdôležitejšie služby na vytvorenie kompletného automatizovaného systému pre správu firemného biznisu [5]. To je dosiahnuté použitím SOA na reprezentovanie biznis funkcií ako jednoduchých, generických rozhraní založených na množine dátových štandardov, ktoré boli pôvodne vyvinuté k internetovej komercii. BPM sa používa k vytvoreniu automatizovaných biznis procesov orchestrovaním biznis funkcií (služieb). A navyše vplyvom pravidiel vrámci BPM alebo oddeleného enginu pravidiel (pripojený k BPM ako služba), biznis môže ponúknuť presnú službu, zväzok produktov, alebo platbu za produkty vrámci definovaného procesu.

Business Process Management je zhluk tokov informácií, ktoré požaduje biznis aktivita k úspešnému ukončeniu transakcie. Definuje pracovné toky, ktoré obsahujú všetky kroky a služby potrebné k danej úlohe. V mnohých prípadoch je informácia získaná, pridaná alebo menená z vnútorného zdroja systému. Niekedy však musí systém založený na SOA siahnuť po iných službách, napríklad externej databázy, aby dokončil svoju prácu.

Namiesto zamerania na presné funkcie, procesy alebo dokumenty, ktoré sú ich súčasťou, sa SOA zameriava na informáciu samotnú. V bankovom procese to môže byť napríklad, informácia o kontaktných údajoch zákazníka. I keď danú informáciu požadujú rôzne oddelenia v rámci banky, jedná sa v podstate stále o tú istú funkcionalitu. Klasickým spôsobom by sme museli do každého systému banky napísať kód v závislosti od typu systému, ktorý by tieto údaje získal. Architektúra založená na SOA to môže dosiahnuť jednoducho, bez ohľadu na funkciu ktorá práve prebieha.

To je umožnené biznis manažérom, ktorý vytvára SOA pracovný tok (work flow) založený na informácií, ktorú potrebujeme v ľubovolnom bode procesu. Poskytuje veľa možností pre zvýšenie efektivity v rozsiahlych systémov. Takisto umožňuje bankám pristupovať k službám publikovaným vonkajšími poskytovateľom, pridaním novej funkcionality bez náročnej a drahej implementácie.

Mnoho starších systémov sa v súčastnosti javí ako webové služby, a tak je oveľa jednoduchšie zdielať dáta medzi nimi a novším softvérom. SOA je veľmi flexibilná a umožňuje užívateľom zmeniť volanú službu v transakcií rýchlo a jednoducho.



Obr. 6: BPM engine [6]

5.3 Enterprise Service Bus

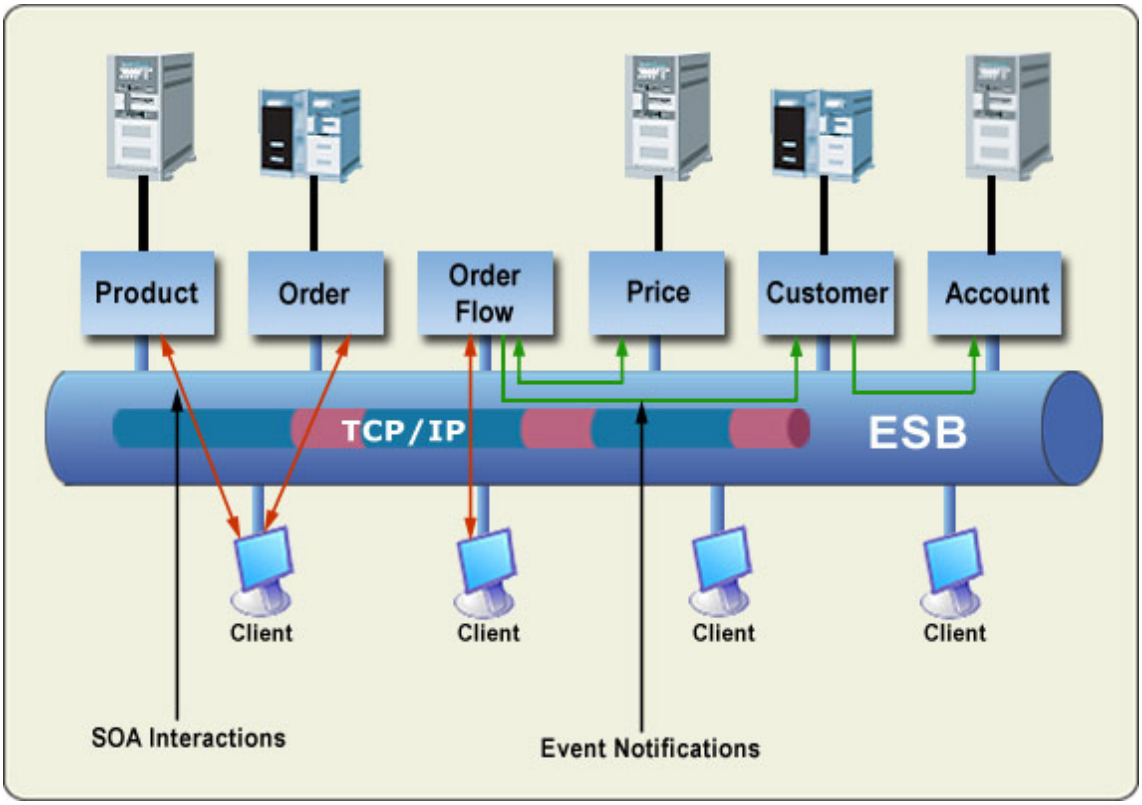
ESB je softvérová architektúra poskytujúca dôležité služby pre komplexnú architektúru pomocou udalosti-riadeného a na štandardoch-založeného enginu správ (zbernica). Vývojári typicky implementujú ESB použitím technológií založených na rozpoznávaní štandardov.

ESB predstavuje abstraktnú vrstvu, ktorá zastrešuje implementáciu rozsiahleho systému správ, ktorá umožňuje integrátorom tejto architektúry využit' hodnotu správ bez písania kódu.

ESB prináša koncept založený na toku dát do prostredia servisne orientovanej architektúry. Predstavuje takisto abstrakciu koncových bodov. Tím umožňuje flexibilitu v transportnej vrstve a slabšie ale jednoduchšie spojenie medzi službami.

Enterprise Service Bus zastáva podobnú funkciu ako klasická zbernica v PC, ale na vyššej úrovni abstrakcie. V rozsiahlych architektúrach potom aplikácie komunikujú cez zbernicu (ESB), ktorá koordinuje tok správ. Výhody spočívajú v menšom množstve pripojení bod-bod, ktoré umožňuje aplikáciam komunikovat'a vedie k jednoduchším úpravám.

Zo softvérového hľadiska ESB reprezentuje softvér, ktorý sa nachádza medzi biznis aplikáciami a umožňuje komunikáciu medzi nimi. Výhodne je ak sa podarí nahradit' všetky priame spojenia pomocou zbernice. To dosiahne zabalením funkcionality jednotlivých súčastí tak, aby boli komunikačne neutrálne. Používa k tomu rôzne modely, ktoré definujú štandardnú množinu správ pre príjem a vysielanie. ESB teda prijme správu, pre-



Obr. 7: ESB [7]

transformuje ju do podoby, ktorú očakáva model daného komponentu a pošle cieľovej aplikácií.

Výhody:

- Rýchlejšie a lacnejšie prispôsobenie existujúcich systémov
- Zvýšenie flexibility, jednoduchosť zmien
- Založené na štandardoch
- Použiteľné od malých po enterprise riešenie
- Preddefinované servisné typy
- Neobsahuje centrálny komunikačný prvok, nehrozí pád celého systému

6 Platformy a rámce orchestrácie webových služieb

Webové služby sú založené na nezávislosti od architektúry, použitej platformy, či implementačnej technológie. Táto nezávislosť je len z hľadiska špecifikácie služby vzhľadom na kompatibilitu s „okolitým svetom“. Konkrétna implementácia softvérov využívajúcich webové služby je plne závislá na implementačnej platforme a rôznych obmedzeniach.

6.1 Sun Java Enterprise Edition

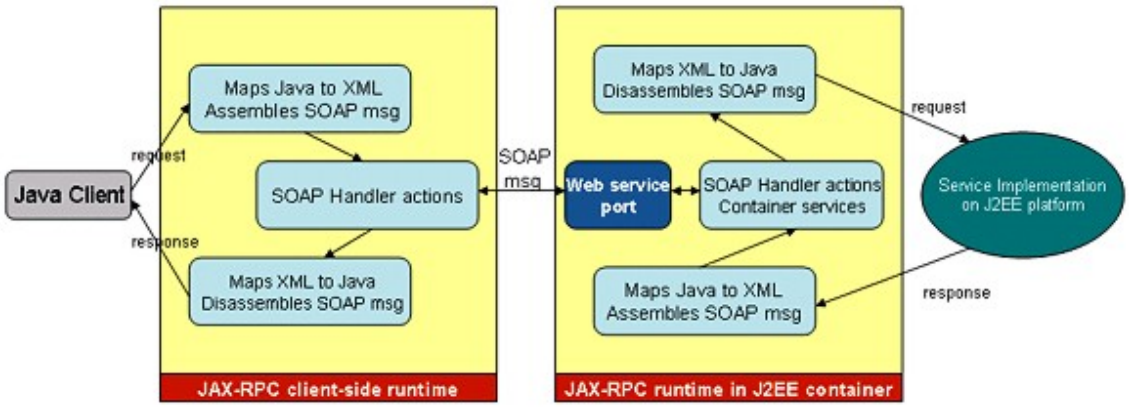
Ide o jednu z najrobustnejších platforiem vôbec, pričom na základe Javy a jej knižníc sú postavené ďalšie. Cieľom je poskytnúť čo najjednoduchšie a rozsiahle prostriedky pre tvorbu softvéru založenom na webových službách.

J2EE bola zameraná na implementáciu jednotlivých technológií webových služieb pomocou základného balíku **WSDP** (Java Web Services Developers Pack). Http servlet predstavuje implementáciu služby. Ten prijíma, spracúva požiadavky a pomocou knižníc pre protokol SOAP a formáty XML analyzuje a interpretuje volanie webovej služby. Následne dáta predá príslušným objektom, ktoré v svojich metódach implementujú požadovanú funkcionalitu.

Balík **WSDP** obsahuje tieto knižnice:

- *Java API for XML Processing* (JAXP) – implementuje triedy pre prácu s XML (tj. SAX, DOM a XSLT).
- *Java API for XML Messaging* (JAXM) – slúži na posielanie a prijímanie asynchrónnych správ v podobe XML dokumentov. Obsluha a kódovanie správ sú založené na špecifikácií *ebXML Transport, Routing, and Packaging specification*.
- *SOAP with Attachments API for Java* (SAAJ) – knižnica pre vytváranie a spracovanie správ v súlade so špecifikáciou SOAP 1.1, vrátane implementácie SOAP príloh.
- *Java API for XML-based RPC* (JAX-RPC) – implementuje vzdialené volanie procedúr (RPC) založené na XML podle SOAP 1.1, obsahuje i nástroj pre prácu s *Web Services Definition Language* (WSDL). JAX-RPC API skrýva komplexnosť správ SOAP pred aplikačným vývojárom. S JAX-RPC programátor správu negeneruje ani nerozoberá.
- *Java API for XML Registries* (JAXR) – knižnica umožňuje vývojárom implementovať podporu prístupu k XML registrom, ktoré sú používané pre uchovanie informácií o publikovaný webových službách, najčastejšie používaný pre *Universal Description, Discovery, and Integration* (UDDI) registri.

Ďalšou knižnicou založenou na platforme JavaEE je **Java API for XML Web Services** (JAX-WS). Pre realizáciu XML webových služieb používa podporné knižnice z balíku WSDP , tie poskytujú prenos XML správ SOAP protokolom cez HTTP, vytvárajú a prijímajú „message-oriented“ a „RPC-oriented“ volanie webových služieb. Nezávislosť na platforme je dosiahnutá vďaka dodržaniu štandardov podpornými triedami a špecifikácií webových služieb v JAX-WS. To umožňuje komunikovať i zo službami, ktoré nie sú napísané v jazyku Java.



Obr. 8: JAX-RPC [17]

6.1.1 Open ESB

Open enterprise service bus je na Jave založené riešenie firmy Sun Microsystems. Open ESB môže byt' použitá ako platforma pre enterprise aplikácie i SOA. Skladá sa z Java Business Integration (JBI), ktorej úlohou je spájat' rozdielne typy zdrojov štandardným a prirodzeným spôsobom. Základnym stavebnym prvkom ESB je BUS (Zbernica), ktorá má na starosti prenos a smerovanie správ správnym komponentom. Komponenty zabezpečujú prijímanie správ (jednotného formátu) zo "Zberniceä ich následné odoslanie ďalšiemu softvéru vo forme, ktorú vyžaduje. Poslednou časťou je Service Engine, umožňujúci pripojenie "Zbernice"k iným kontajnérom (J2EE kontajnér). Ten transformuje správy a smeruje požadované aktivity.

Open ESB nájdeme integrované v Netbeans IDE 6.x (spolu s BPEL, XSLT, XSD a WSDL editorom) a vyššie a aplikačnom servery Glassfish 2.x a vyššie. Hlavným problémom tohto riešenia je, že po prevzatí Sun Microsystems firmou Oracle, bola podpora tohto projektu zrušená.

6.1.2 Java Business Integration

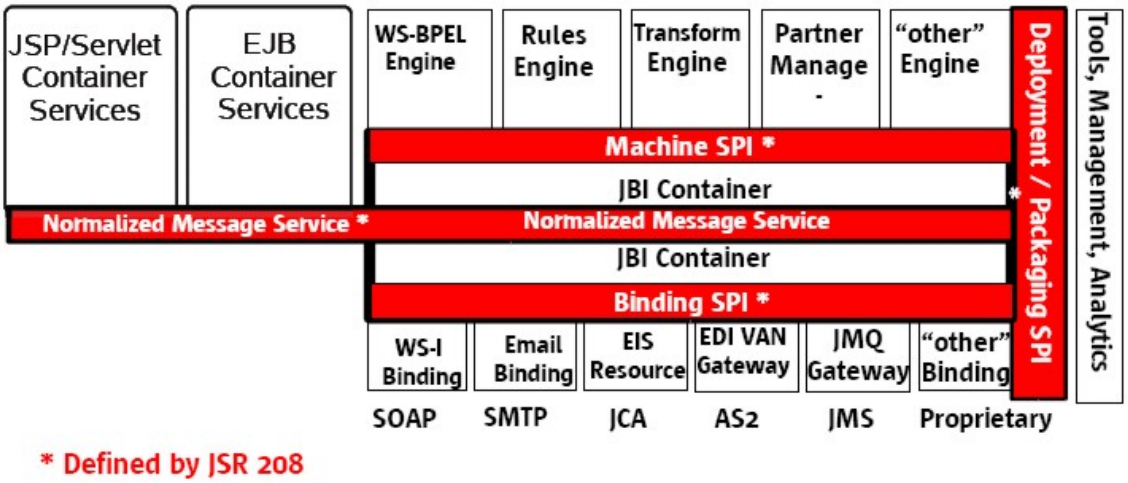
Java Business Integration (JBI) je špecifikácia pre štandard, ktorý popisuje „plug-in" technológiu pre softvérové systémy založené na SOA a integráciu serverových softvérov. JBI prevzalo servisne orientovanú architektúru aby dosiahlo maximálne oddelenie medzi komponentmi a vytvorilo dobre definovanú sémantiku založenú na štandardizovaných správach. Poskytuje rozhranie služby (SPIs), ktoru engine pripojil a normalizovanú službu správ na komunikáciu medzi jednotlivými komponentami.

Výhody:

- Servisne orientovaná architektúra znamená väčšiu flexibilitu, rozšíriteľnost' a škálovateľnost'.
- Implementácia služieb pripojených do JBI nie je závislá na programovacom jazyku.

- Pridanie nového enginu prebieha priradením do SPI a definovaním typu správ.
- Vďaka vyššie uvedeným bodom si zákazník môže vybrať najlepšie dostupné riešenie.

Príklad architektúry JBI (JSR 208 od firmy Sun)



Obr. 9: JBI [17]

6.2 JBoss Enterprise SOA a Data Service Platform

Spoločnosť Red Hat sa zaoberá enterprise riešeniami už dlhšiu dobu. Vo svojom balíku JBoss® Enterprise Middleware (napísané v jazyku Java, štandard J2EE) ponúka „open source" portfólio pre SOA.

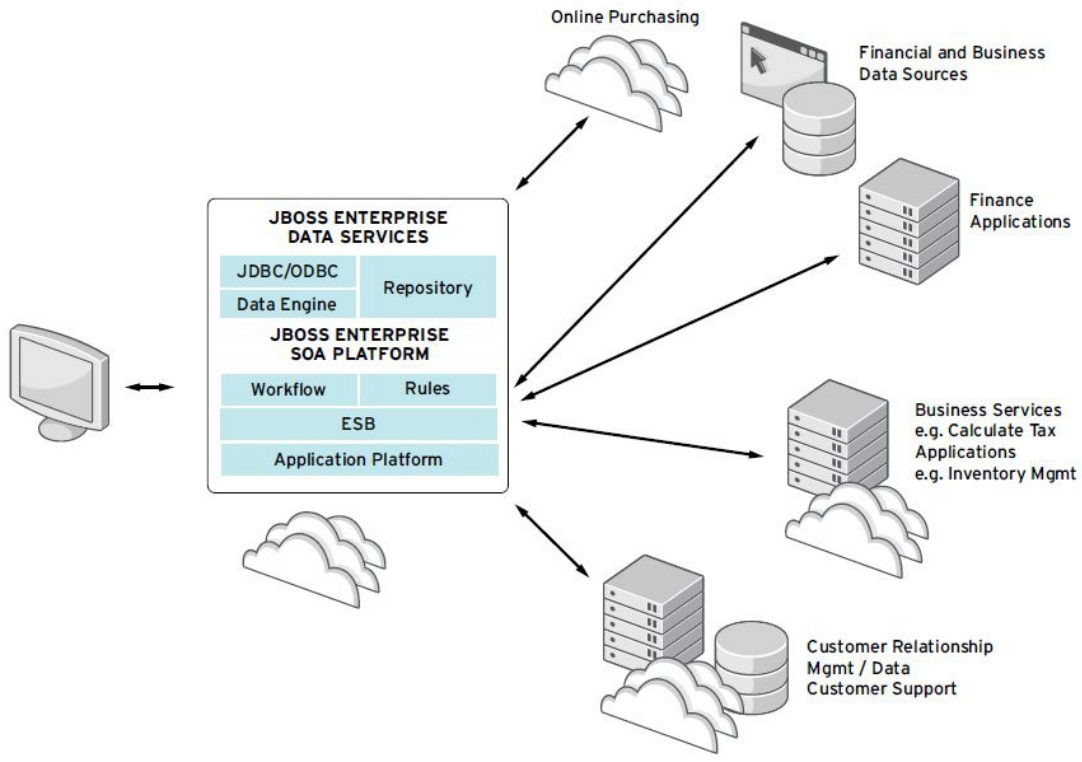
V roku 2008 Red Hat vydal rozšírenie JBoss Enterprise SOA Platform, založenom na novej generácií ESB (Enterprise Service Bus) [18]. Tá zabezpečuje SOA integráciu, aplikačnú integráciu a platformu pre orchestráciu procesov. Hlavná výhoda tohto open source riešenie spočíva v sofistikovanom použití SOA a automatizovanom nasadení biznis procesov. Jedná sa o EDA (Event-Driven architecture), ktorá poskytuje lepšiu odozvu pri riadení biznis udalostí. Táto platforma v sebe integruje ESB, pravidlá orchestrácie, automatizáciu biznis procesov a JBoss Enterprise Application Platform.

V roku 2011 pribudla k JBoss Enterprise SOA Platform nástavba JBoss Enterprise Data Services Platform, ktorá priniesla virtualizáciu dát a integračné možnosti pre enterprise riešenie. Jedná sa o sadu nástrojov a komponentov na podporu integrácie a použitia dát (z viacerých zdrojov) aplikáciami a biznis procesmi. **Obsahuje:**

- mechanizmy pre vytváranie pohľadov na dáta (data views), ktoré sú prístupné pomocou štandardných protokolov
- adresár metadát
- „runtime“ prostredie zabezpečujúce enterprise výkon, integritu dát a bezpečnosť.

Súčastnostným prístupom k viacerým dátovým zdrojom (súbory, aplikácie, databázy, služby, . . .) a ich poskytovaním v jednotnej forme odbremení aplikácie od potreby znalosti detailov každého dátového zdroja. S JBoss Enterprise Data Services Platform sú dáta prístupné v reálnom čase, bez potreby kopírovania alebo presunu na iné miesto. Prístup k dátam je možný cez JDBC, ODBC, alebo sú publikované ako služba v UDDI registri pomocou ESB.

Na obrázku 10 je vidieť prácu JBoss Enterprise SOA Platform a JBoss Enterprise Data Services Platform

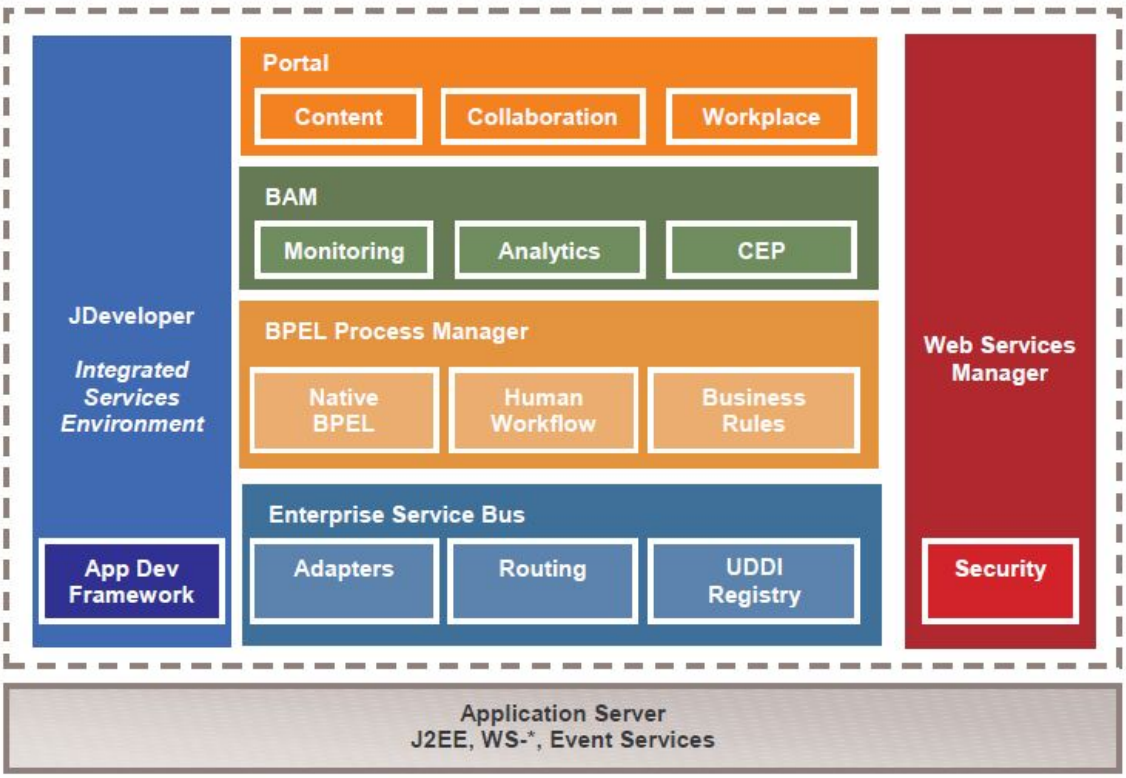


Obr. 10: JBoss SOA [18]

6.3 Oracle SOA Suite

Oracle SOA Suite je základnou technológiou Oracle Application Integration Architecture: Oracle riešenia poskytovania aplikačnej integrácie. Ide o komplexné riešenie pre spájanie, vývoj a riadenie servisne orientovanej architektúry. SOA Suite umožňuje maximalizovať znovupoužitie existujúcich IT štruktúr, technológií a komponentov, nezávisle od prostredia (OS, aplikačný server, . . .).

Oracle Jdeveloper, Oracle Application Development Framework (Oracle ADF) a **Oracle TopLink** sú hlavné vývojové komponenty Oracle SOA Suite, ktoré vytvárajú integrované servisné prostredie pre tvorbu, registráciu a spracovanie rôznych typov rozhraní. Komponenty Oracle SOA Suitu môžete vidieť na obrázku nižšie.

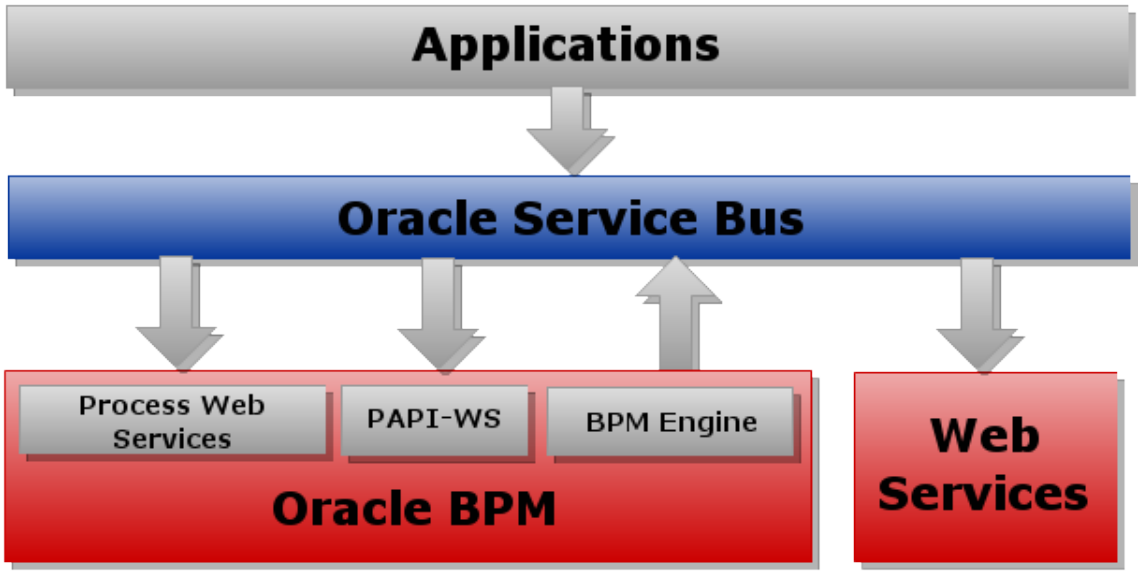


Obr. 11: Oracle SOA Suite [14]

6.3.1 Oracle BPM a OSB

Tieto dva produkty sú dôležitou súčasťou SOA prostredia. Oracle Service Bus (OSB) môže poskytnúť agregáciu, transformáciu a bezpečnosť služieb, zatiaľ čo Oracle Business Process Management (OBPM) umožňuje orchestráciu služieb, ktoré OSB vystavuje. Oracle Service Bus predstavuje ralizáciu SOA a Event Driven Architecture (EDA), čím sú distribuované aplikácie integrované pomocou slabo-prepojeným paradigmatom. Exis-

tuje niekoľko kombinácií ako použiť tieto dva komponenty. Záleží od konkrétnej potreby biznisu a zvolenej architektúry. Príklad použitia môžete vidieť na obrázku.



Obr. 12: Oracle BPM a OSB [4]

Medzi kľúčové schopnosti OSB patrí:

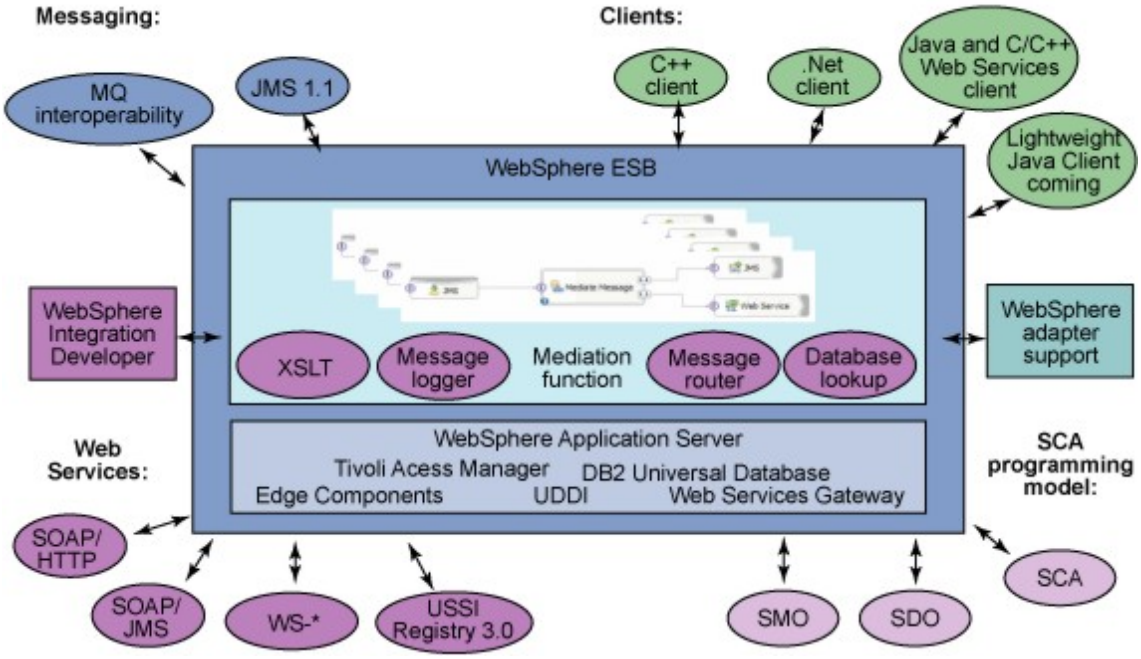
- *Multitransportná zbernica* – na základe štandardných protokolov ako SOAP, HTTP(s) alebo JMS
- *Komplexná transformácia biznis dát*
- *Infraštruktúra pre riadenie a nasadenie komponent biznisu*
- *Enterprise konenktivita* – pomocou vstavaných adaptérov pre databáze, Oracle AQ, JMS, email, FTP a ďalšie služby. Takisto obsahuje adaptéry J2EE Connector Architecture (JCA).
- *Flexibilné smerovanie* – pomocou filtrov na základe obsahu.

6.4 WebSphere

Firma IBM prišla na trh s riešením založeným na Java nástrojoch – WebSphere. To umožňuje vytvárať sofistikované webové stránky a systémy. Základom je WebSphere Application Server (WAS), aplikačný server poskytujúci platformu pre beh Java web aplikácií a služieb. Namiesto klasických rozhraní (CGI) používajú stránky Java servlety, ktoré sú rýchlejšie. WebSphere podporuje rozhrania otvorených štandardov ako Common Object Request Broker Architecture (CORBA) a Java Database Connectivity (JDBC) a je navrhnuté pre beh nezávislé od systémovej platformy. Jednotlivé edície sú prispôsobené potrebám zákazníkov, a to od malých firiem po veľké biznis organizácie. WebSphere obsahuje Studio, vývojové prostredie s množstvom doplnkov.

6.4.1 Websphere ESB

WebSphere Enterprise Service Bus (WebSphere ESB) poskytuje hosting pre integráciu aplikácií a dát ako služieb. Je navrhnuté na vývoj systémov založených na servisne orientovanej architektúre. Napomáha vytvárať nové služby a pripájať ich poskytovateľov pre jednoduché vytvorenie biznis systémov. Umožňuje virtuálne pripájať ľubovolnú biznis aplikáciu a podporuje servisné brány, prekladanie a výber služieb ako aj pravidlá pre dynamické SOA riešenia. WebSphere ESB prináša konzistenciu do „point-to-point“ spojení a redukuje IT komplexnosť, tak aby sme mohli zvýšiť efektívnosť biznisu.



Obr. 13: WebSphere ESB [9]

WebSphere ESB je ideálne ak máte ďalšie riešenia na WebSphere Application Server, WebSphere Portal, WebSphere Commerce, alebo BPM platforme. Môžete dosiahnuť efektivitu v nasadení, cene, a čase naprieč midlevér produktom, pridaním WebSphere ESB do Java IT prostredia. WebSphere ESB je distribuované ako súčasť IBM Business Process Manager i ako samostatný produkt.

IBM WebSphere ESB:

- Prináša konzistentnosť do komunikácie medzi rôznymi komponentami
- Poskytuje inteligentnú konektivitu na internetovú aplikačnú štruktúru ľubovolným aplikáciam a dátam.
- Podporuje: MQ a JMS messaging, HTTP, EJB, databázy, súbory, prenos dát, email, Lotus Domino, CICS, IMS, SAP, Oracle,...
- Jednoducho rozširuje IBM Business Process Manager o orchestráciu služieb a BPM.
- Integruje WebSphere Service Registry a Repository pre SOA riešenie.

6.5 Microsoft .NET

Firma Microsoft má takisto zastúpenie v oblasti SOA architektúry a to v rámci platformy .NET. Potrebné knižnice obsahuje balík .NET Framework SDK, ktorý je dobre integrovaný v nástroji Visual Studio .NET. Samotné webové služby sú potom implementované pomocou ASP.NET. Konzumenti webových služieb sú podporovaní v balíku *Microsoft Office* pomocou *Office Toolkit for Web Services*.

Platforma .NET využíva XML Web služby, ktoré sú založené na štandardoch internetových protokolov a komponentového vývoja aplikácií. Reprezentujú skrytú funkcionalitu a tá môže byť volaná bez závislosti na jej konkrétnej implementácií. V *Microsoft® BizTalk™ Server* vytvárame XML Web služby pomocou Microsoft SOAP Toolkit 2.0 a Microsoft Visual Studio® .NET.

BizTalk Server je časťou Microsoft Windows Server System a poskytuje následujúcu funkcionalitu:

- Integrácia na úrovni správ, z enterprise prostredia (Enterprise Application Integration - EAI) do Internetu (business-to-business, or B2B), pomocou BizTalk Server Messaging.
- Automatizáciu biznis procesov použitím BizTalk Server Orchestration služieb, ktoré poskytujú možnosť implementovať dlhodobé a slabo prepojené biznis procesy.

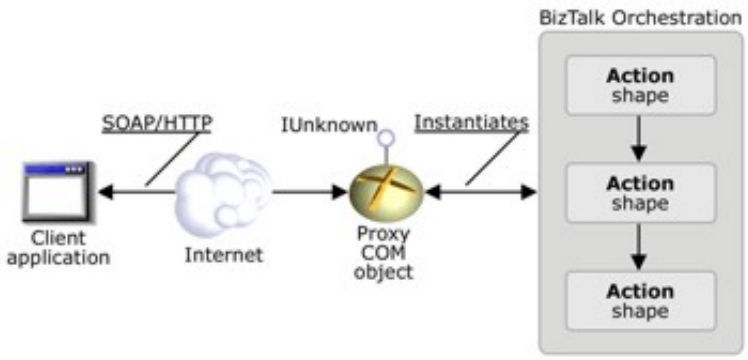
6.5.1 BizTalk Orchestration

BizTalk Orchestration prináša pre vývojárov spôsob ako vytvoriť škálovatelné a dobre dostupné webové služby. Orchestrácia je proces vytvorený v Microsoft Visio®-based BizTalk Orchestration Designer, serializovaný v XML, spustený a riadený COM+ službami (XLANG Scheduler). Tento prístup so sebou prináša určité problémy ktoré treba riešiť. Ide

o interakciu medzi webovými službami, transakčný manažment, odchytávanie výnimiek, konkurenčnosť a interakcia so aplikáciami bez podpory XML. Preto okrem COM komponentov BizTalk Orchestration Designer umožňuje volať služby Script komponentami, frontami správ alebo BizTalk správami.

A BizTalk Orchestration môže byť vytavená ako XML Web služba tromi spôsobmi [11]

- Programovo
- Použitím SOAP Toolkit 2.0
- Použitím Visual Studio .NET



Obr. 14: Vystavenie BizTalk Orchestration pomocou COM objektu [11]

Porovnanie jednotlivých platforiem vidíte v prílohe A na obrázku 27. Parametre, ktoré som zvolil, sú potrebné pre správny beh aplikácie v prostredí SOA a riadené BPELom. Hlavným aspektom pri rozhodovaní bude asi cena a vlastníctvo iných balíkov či komponent danej firmy. Pri voľbe platformy totiž nezohrávajú úlohu jazkyky a rámce, použité pri implementácii služieb.

7 Zadanie

Zákazník požaduje informačný systém, ktorý bude udržovať evideniciu skladových zásob, predajov a nákupov rôznych materiálov a tovaru. Mal by pokryť oblasť naskladnenia, výroby a fakturácie hotových výrobkov.

7.1 Vízia

Vo firme (obchod, sklad, reštaurácia,..) potrebujeme viesť evidenciu skladových zásob, množstvo nakúpeného a spotrebovaného materiálu za určité obdobie, cenu a celkové náklady za materiál, zoznam dodávateľov. Tiež musíme evidovať doklady o predaji tovaru. Všetky tieto informácie sú potrebné pre daňové priznanie, zo zákona povinné výkazy, i pre vlastnú potrebu, aby mal majiteľ prehľad o tržbách.

7.2 Cieľ

Prevádzkovateľ firmy je zo zákona povinný poskytnúť zákazníkovi doklad, na ktorom je zoznam predaného tovaru, jednotková cena s DPH, celková cena s a bez DPH, dátum a názov firmy či živnostníka so základnými obchodnými údajmi (IČO, DIČ, adresa).

Dôležité je mať prehľad o tržbách, aby majiteľ vedel, či je firma zisková alebo stratová. Na základe toho môže upraviť ceny, zmeniť ponuku výrobkov a materiálov.

Tabuľky so skladovými zásobami a spotrebou jednoducho zobrazia užívateľovi, ktorý tovar je vodné doobjednať a zo zoznamu dodávateľov môže zvoliť najlacnejšieho.

Možnosť búdúceho prepojenia s účtovníctvom (daňové priznanie, zákonom nutná evidencia, výkazy) predstavuje dôležitú vlastnosť systému z pohľadu zákazníka.

Dôvodom vytvorenia celého systému je zdĺhavosť a komplikovanosť ručnej správy faktúr, dokladov a evidencie skladových zásob. Cieľom je priniesť zrýchlenie, zautomatizovanie a zefektívnenie procesov danej firmy.

8 Špecifikácia požiadaviek

Definujeme role a biznis prípady použitia, ktoré budú informačným systémom pokryté. Kedže sa jedná o protoyp, popis procesov nie je detailný, ale predstavuje základný náčrt funkcionality.
Role užívateľov:

- Zákazník (User)[U] - najnižšie práva, zatiaľ len pasívny užívateľ
- Obsluha (predajca) [O] – možnosť vytvoriť (vložiť zoznam tovaru s cenami a množstvá) a vytlačiť doklad pre zákazníka.
- Účtovník [K] – na základe množstva nakúpeného, spotrebovaného materiálu a tržieb (denné a mesačné uzávierky) z predaja dokáže poskytnúť všetky informácie o stave a fungovaní firmy (kontroly alebo nadriadený). Pridáva nový tovar, nákupy, firmy a komponentáre.
- Admin (zodpovedný vedúci) [A] – správa užívateľov a systému, celkový prehľad skladov a výstupov

Hlavný biznis proces systému:

1. Účtovník (skladník) príjme tovar od určitej firmy. V zozname firiem vyberie existujúcu alebo vytvorí novú. Pridá nový nákup a naplní ho materialmi a množstvami s nákupnou cenou. Po uložení sa aktualizujú skladové zasoby daného tovaru na jednotlivých skladoch.
2. Účtovník definuje, ktorý tovar je predajný a zobrazí sa konečnej obsluhe. Ďalej vytvára zoznam komponent, z ktorých môže byť daný výrobok zhotovený (zložky jedla v reštaurácií, súčiastky prístroja).
3. Obsluha výtvorí doklad a pridá položky s predávaným tovarom. Po ukončení sa doklad uzavrie a zo skladu sa odpíše materiál na základe komponent tovaru.
4. Zákazníkovi je odovzdaný doklad.

8.1 Biznis prípady použitia

Užívatel systému:

- 1.1 Prihlásenie
- 1.2 Odhlásenie

Obsluha:

- 2.0 Nový doklad
- 2.1 Vytlač doklad/uzatvorenie dokladu (možno blok/faktúra)

- 2.2 Storno dokladu
- 2.3 Zobraz doklady
- 2.4 Odstrániť doklad
- 2.5 Vložiť položku dokladu
- 2.6 Odstrániť položku dokladu
- 2.7 Zmeniť položku dokladu

Účtovník:

- 3.0 Pridať materiál
- 3.1 Odstrániť materiál
- 3.2 Upraviť materiál (nastavit na predajný)
- 3.3 Pridaj komponentu
- 3.4 Uprav komponentu
- 3.5 Zruš komponentu
- 3.6 Pridať firmu
- 3.7 Odstrániť firmu
- 3.8 Upraviť firmu
- 3.9 Zobraz firmy
- 4.0 Zobraz sklady
- 4.1 Zobraz materialy na sklade
    - 4.1.1 Predaje materialu
    - 4.1.2 Nákupy materiálu
        - 4.1.2.1 Nákupy materiálu od danej firmy
    - 4.1.3 Spotreba
- 4.2 Zobraz obrat
- 4.3 Nový nákup
- 4.4 Odstrániť nákup
- 4.5 Vložiť položku nákupu
- 4.6 Upraviť položku nákupu

- 4.7 Odstrániť položku nákupu
- 4.8 Zobraz firmu + nákupy

Admin: Celkový prístup a prehlad v celom systéme.

V priloženom dokumente **biznisPripadyPouzitia.pdf** nájdete popis všetkých vymenovaných prípadov použitia a BPEL procesov. Prostredie, ktoré som použil pre vývoj, zároveň umožňuje i modelovanie orchestrácie (otvorením projektov *BPEL NetBeans 6.5.1). Na základe kódu generuje diagram (funguje i opačným smerom). Výhodou je rýchlejší vývoj, prehľadnejší proces a možnosť priamej komunikácie a odozvy od zákazníka.

8.2 Technická špecifikácia

Malo by sa jednať o webovú aplikáciu a systém musí zvládnuť prístup viacerých užívateľov (10-100) v primeranom čase (v závislosti na náročnosti operácie - do 10s). Aplikácia bude bežať na intranete (prístup len zamestnancom spoločnosti) a do budúcnosti je plánováné prepojenie s internetovým obchodom. Dôležitá je univerzálnosť systému a konfigurovatelnosť biznis procesov. Platforma a databáza nie je daná. Pre správnu prácu s aplikáciu je vhodné školenie užívateľov. Dáta by mali byť pravidelne zálohované.

9 Analýza a návrh

Z pohľadu SOA riešenia hovoríme o "Servisne orientovanej analýze a návrhu". Vyžívajú sa štandardné postupy objektovo-orientovanej analýzy a návrhu spolu s ďalšími ramcami a jazykmi pre modelovanie ako sú: **EAf** - Enterprise Architecture frameworks, **BPM** - Business Process Modeling. V mojej práci som sa zameral hlavne na správny návrh webových servis a logiky komunikácie prostredníctvom orchestrácie. Prezentačna vrstva predstavuje len načítanie dát od užívateľa, ich následné predanie orchestru alebo webovej službe a zobrazenie výsledku. I samotné podsystémy predstavujú základné prototypy s jednoduchými objektami a logikou služieb. Pri ich návrhu a rozdelení bol kladený dôraz na čo najmenšiu závislosť a možnosť budúceho rozšírenia.

9.1 Analýza a návrh systému

Servisne orientovaný (SO) a *objektovo-orentovaný* prístup (OO) sa nevylučujú, práve naopak:

- Servisne orientovaný prístup je založený na voľnom previazaní jednotlivých služeb. Naopak OO prístup kladie dôraz skôr na presné definované vzťahy medzi triedami, čo vedie k oveľa pevnejším väzbám entít (objektov).
- Obidva prístupy majú podobný pohľad na abstrakciu základných entit vytvorenú pre komunikáciu. V OO prístupe máme presne definované rozhrania objektov. V SO prístupe tuto úlohu väčšinou plní nejaký druh popisného dokumentu (u webových služieb to je popis služby vytvorený vo WSDL)
- SO prístup predpokladá, že rozsah činností, ktoré daná služba poskytuje, je dynamický. OO prístup definuje objekty, které sú viac špecifické v danom obore pôsobnosti.
- Aktivita služieb je vyvolána až príchodom správy. Správy obsahujú okrem dát i logiku spracovania a určitým spôsobem riadia činnosť služieb. V OO prístupu su dáta zviazaná s logikou do objektov.

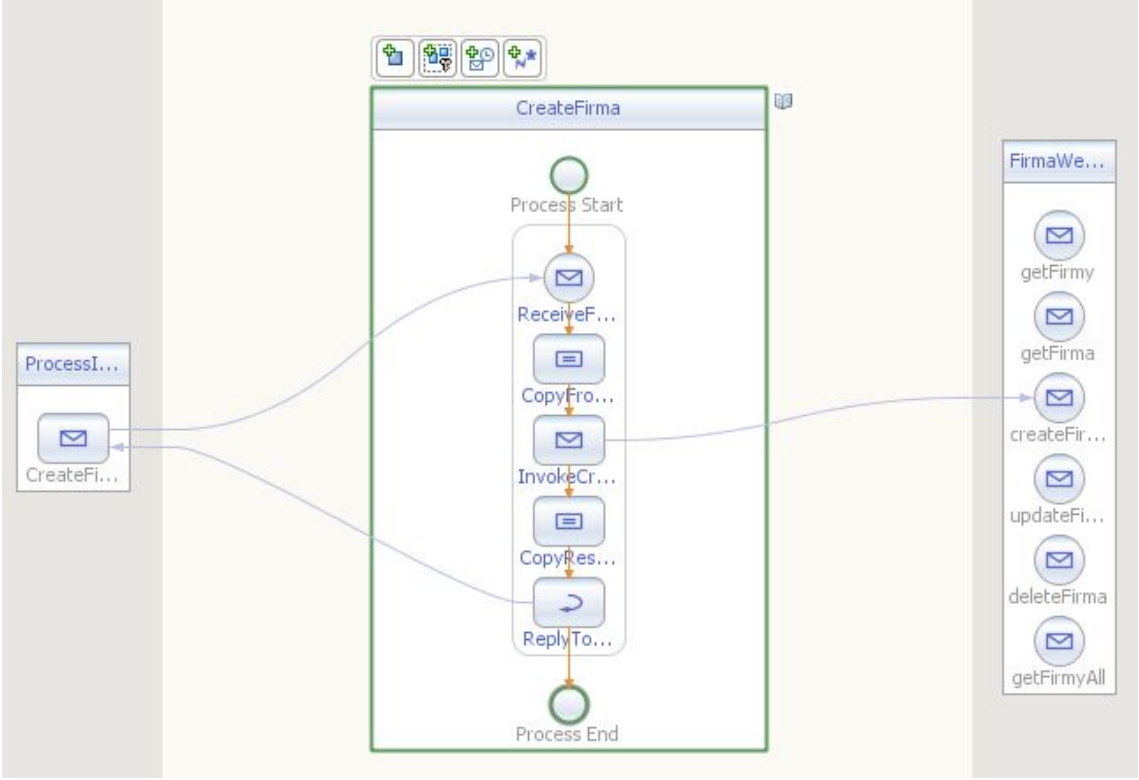
Základná vlastnosť služieb, ktorou sa odlišujú od objektov je ich bezstavovosť. Naopak pri OO prístupe, zapúzdrením logiky a dát vzniká stavová závislosť.

9.2 Business Process Modeling a Enterprise Architecture

Biznis proces je postupnosť činností, ktorých cieľom je realizovať určitý podnikateľský zámer (aktivitu, flow) v organizácií. BPM je balík metód, postupov a prístupov určených k špecifikácií a analýze procesov. Na jednej strane predstavuje určitú abstrakciu procesu, ale zároveň presne popisuje proces a analyzuje jeho chovanie. **BPM** používa niekoľko techník:

- Business Process Modeling Notation (BPMN)
- Event-Driven Process Chains (EPC)
- Business Process Execution Language (BPEL).

V mojej práci som použil BPEL. Tento jazyk špecifikuje biznis proces predovšetkým z hľadiska interakcií jeho častí (napr. webových služieb). Na obrázku 15 vidíte priklad jednoduchého procesu vytvorenia záznamu firmy, volaním webovej služby FirmaWebService a jej metódy createFirma. Modeloval som pomocou BPEL designeru, ktorý je implementovany vo vývojovom prostredí NetBeans 6.x. Zároveň slúží i k realnej implementácií a konfigurácii procesu.



Obr. 15: Príklad BPEL jazyka

Rámec pre podnikovú architektúru (EAf) sa používa pre optimalizáciu a zefektívnenie existujúcich procesov v podniku. Popisuje ich štruktúru a chovanie. Je vhodny pre SOA riešenie, ktoré vznikajú dekompozíciou a integráciou už existujúcich systémov.

9.3 Servisne orientovaná analýza a návrh

Identifikácia a klasifikácia služieb

Používajú sa tri prístupy [2]. Prvým je analýza zhora-dole (nazývaná i doménová dekompozícia). Zaoberá sa biznis procesmi v danom podniku a ich vzťahmi. Procesy sú rozložené na podprocesy a tie na prípady použitia. Zoznam nájdete v kapitole 8 Špecifikácia požiadaviek. Na ich základe môžeme identifikovať potrebné služby a metódy. Následuje analýza zdola-nahor. V tejto fáze sa prechádzajú existujúce riešenia a hľadajú sa vhodni kandidáti na komponety a primitívne služby. Posledným krokom je modelovanie cielových služieb. Odhalíme ním služby, ktoré neboli identifikované v predchádzajúcich krokoch. Taktiež upresním, prípadne redukujeme počet služieb s ohľadom na potreby systému, výkonnosť s ďalšie faktory.

V **SOA architektúre** môžu byť služby tvorené komponentami alebo inými službami. Pre moje potreby budeme používať len primitívne služby, kedže o komunikáciu a jej riadenie sa stará BPEL. Zamedzíme tak neprehladnosti, ktorá by mohla nastať. Samozrejme služby jednej komponenty môžu medzi sebou komunikovať i napriamo, kedže nevystupujú mimo kontextovú doménu svojho modulu. Z môjho pohľadu je, ale najlepšie presunúť celú logiku do BPEL procesu, čím sa vyhneme nejasnostiam a získame lepšiu kontorolu nad tokom dát a prebiahajúcimi operáciami. Dôležité je i nájsť hranicu, kedy použiť BPEL, a kedy nechať logiku na danej metóde služby. Môže nastať prípad, pri ktorom by sa naš proces stal príliš rozsiahli a ťažkopádny na údržbu.

Analýza podsystémov a špecifikácia komponent

Spracovávame biznis prípady užitia vzniknuté domémovou dokompozíciou. Na ich základe vytvoríme objektové modely pre funkčné jednotky a podsystémy. Po návrhu prichádza na radu špecikácia štruktúry a komunikácie komponent.

Zostavenie a realizácia služieb

Každý podsystém je tvorený komponentami, ktoré sprístupňujú jeho funkcie. Z nich následne zostavíme služby. V prípade už existujúceho riešenia sa rozhodujeme či použiť už existujúcu službu, alebo vytvoriť úplne novú.

9.4 Identifikované služby a BPEL procesy

Celú analýzu a navrh som vykonal v niekoľkých iteráciach podľa vyššie uvedeného postupu. Na základe biznis prípadov použitia som identifikoval BPEL procesy a následne metódy služieb, ktoré k tomu potrebujem. Systém je rozdelených do niekoľkých logických podsystémov (komponent), ktoré vystupujú prostredníctvom nasledujúcich služieb:

- **WebModuleSklad** - služby: SkladWebService, SkladovanieWebService. Táto komponenta zaobaľuje funkcionalitu spojenú so skladovaním materiálov, aktualizáciou množstva a evidenciou zásob.
- **WebModuleMaterial** - služby: MaterialWebService, ZlozkaWebService. Modul slúži pre správu materiálov, ich koponent a cien pre predaj a nákup.

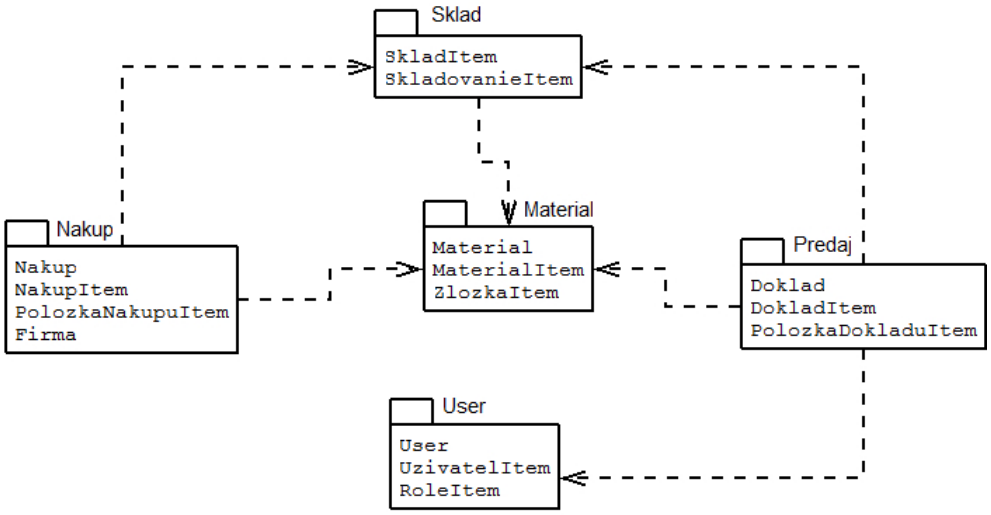
- **WebModuleNakup** - služby: FirmaWebService, NakupWebService, PolozkyNakupuWebService. Funkcie pre prácu s nákupmi od rôznych dodávateľov.
- **WebModulePredaj** - služby: DokladWebService, PolozkyDokladuWebService. Komponenta poskytujúca prácu s dokladmi a predajom tovaru.
- **WebModuleUser** - služby: UserService. Evidenicia užívateľov.

Podrobný popis webových služieb nájdete v jednotlivých moduloch (vytvorené ako samostatné webové aplikácie) pod záložkou: Web services.

Identifikované BPEL procesy:

- Z oblasti práce s materiálom: Create, Update a Delete materiálu.
- Zamerané na predaj: Create, Update a Delete Dokladu a jeho položiek, uzatvorenie a storno dokladu + aktualizácia stavu na sklade.
- Procesy nákupu materiálu: Create, Update a Delete Dokladu a jeho položiek, update množstva tovaru materiálu na sklade.
- Ekonomické procesy - nákup, spotreba, obrat za určíté obdobie.
- Správa užívateľov

Komunikácia komponent systému pomocou webových služieb je zobrazená v konkrétnej implemntácií každého BPEL procesu. V priloženom dokumente **biznisPripadyPouzitia.pdf** nájdete ich slovný popis. Na obrázku 16 môžete vidieť vyobrazenie závyislosti medzi podsystemami.



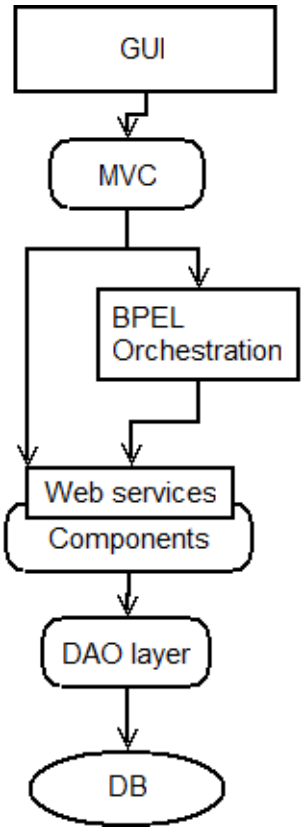
Obr. 16: Komponenty

Triedy samotné obsahujú len `get` a `set` metódy. Logika pre prácu s objektami a ich plnenie sa nachádza v "Service"triedach, ktoré predstavujú implementáciu metód webových služieb. Triedne diagramy modulov nájdete v prílohe A

Prezentačná vrstva bude dodržiavať zásady MVC modelu, pričom triedy forms predstavujú objekty pre prácu s formulármi a v action metódach voláme webové servisy a Composite application s BPELom. Jednotlivé časti systému sú prepojené, tak ako bolo popísané v časti 5.3.

9.5 Architektúra aplikácie

Systém rozdelíme na moduly, ktoré s okolím komunikujú prostredníctvom webových služieb. Kooperácia (spolupráca modulov, predávanie dát) bude riadená pomocou orchestru. Zobrazenie výstupov a interakciu s užívateľom sprostredkuje webová aplikácia. Tá bude zasielať požiadavky o dáta a operácie buď priamo mudulom, alebo v prípade zložitých biznis procesov orchestru. Ten predstavuje konfiguráciu a biznis logiku. Architektúru systému znázorňuje následujúci obrázok 17.



Obr. 17: Architektúra systému

9.6 Databázový návrh

Dáta budeme evidovať pomocou následujúcich tabuliek:

- Material - informácie o materiálu/tovare
- Zlozka - zoznam komponent materiálu s množstvami
- Firma - dáta o dodávateľoch
- Doklad - suma, dátum, informácie pre kupujúceho
- PolozkaDokladu - položky predaného tovaru
- Nakup - nákupy od materiálu od firiem
- Sklad - informácie o skladoch
- Skladovanie - priradenie materiálu k danému skladu, s aktuálnym množstvom
- User - údaje o užívateľoch systému (v budúcnosti o nakupujúcich)
- Role - práva prístupu do systému
- Funkcia - previazanie User-Role
- Stavdokladu - číselník
- Typdokladu - číselník

Štruktúru kompletnej databáze môžete vidieť v prílohe na obrázku 28. V priloženom dokumente **datovySlovnik.pdf** nájdete podrobný dátový slovník.

10 Implementácia

Informačný systém je implemntovaný v jazyku Java a BPEL. Použil som vývojové prostredie NetBeans 6.5.1 s pluginmi pre podporu SOA. Celá aplikácia beží na aplikačnom serveri GlassFish 2.1, ktorý musí mať nainštalovanú súčasť "Composite Applications". Tá obsahuje Open ESB V2 projet s JBI komponentami:

- *BPEL service engine* - podporuje BPEL 2.0 choreografiu služieb spolupracujúcich v rámci kompozitnej aplikácie.
- *Java EE service engine* - chová sa ako premostenie medzi aplikačným serverom a JBI implementáciou. Webove služby definovaná v aplikakačnom servry sú automaticky vystavené na ESB (Enterprise Service Bus) v prostredí JBI. Ďalej môžeme volať poskytovateľov služieb, ktorý sú dostupný skrz ESB.
- *HTTP/SOAP binding component* - poskytujú prístup externým poskytovateľom a príjemcom webových služieb.

GlassFish je súčasťou NetBeansu, čo bol dôvod, prečo som zvolil práve toto prostredie. Umožnuje jednoduchú implementáciu a nasadenie všetkých potrebných súčastí môjho informačného systému. Taktiež tu nájdete BPEL editor pre vizuálne modelovanie a implementáciu celého procesu vrátane logiky volania a spracovania správ webových služieb. Databáza beží na softvére od firmy Oracle.

10.1 Technológie použité pri implementácii

Každy modul má **DAO** vrstvu implementovanú pomocou Spring JDBCtemplate. Tá by mala zamedziť nesprávnemu použitiu databázových zdrojov a optimalizovať dotazy.

Pre tvorbu webových služieb som použil **JAX-WS** knihovnu. Poskytuje prostriedky pre realizaciu XML webových služieb a ich klientov na platforme Java EE. Využíva k tomu podporné knihovny z balíka WSDP, tie zabezpečujú prenos XML správ SOAP protokolom cez HTTP, vytvárajú a prijímaju „message-oriented" a „RPC-oriented" volania webových služieb. Vďaka dodržiavaniu štandardov a špecifikácii pre webove služby, je JAX-WS nezávisla na platforme.

Implemenácia **poskytovateľa** webovej služby je realizovaná pomocou anotácii v Java triede. WSDL popis a oblužny kód naviazaný na knihovnu JAX-WS je následne vygenerovaný automaticky. Webová služba je realizovana ako rozhranie (trieda) s anotáciou `@WebService` a pre jej metódy použijeme `@WebMethod`. Typy parametrov a návratových hodnôt sú obmedzené JAXB (Java Architecture for XML Data Binding). Samozrejme môžete použiť i vlastné typy. Príklad služby MaterialWebService a metódy createMaterial:

```
@WebService()
public class MaterialWebService {

    /**
     * createMaterial
     */
```

```
    @WebMethod(operationName = "createMaterial")
    public Material createMaterial(@WebParam(name = "material")
    Material material) throws MaterialException{
        try {
            MaterialServiceImpl service = new MaterialServiceImpl();
            return service.createMaterial(material);
        } catch (ApplicationException ex) {
            throw new MaterialException(ex);
        }
    }
}
```

Výpis 2: Poskytovateľ služby

Spotrebiteľa vytvoríme pomocou JAX-WS, ako lokálnu proxy pre vzdialenú webovú službu pomocou anotácie `@WebServiceRef`. Tá má rovnaké rozhranie ako trieda implementujúca službu u poskytovateľa. JAX-WS RI vygeneruje všetky potrebné objekty pre jej správne volanie. Samotná trieda obsahuje definíciu endpointu s umiestnením WSDL dokumentu. Príklad časti rozhrania:

```
@WebService(name = "MaterialWebService", targetNamespace = "http://webservice/")
@XmlSeeAlso({
    ObjectFactory.class
})
public interface MaterialWebService {

@WebMethod
    @WebResult(targetNamespace = "")
    @RequestWrapper(localName = "createMaterial", targetNamespace = "http://webservice/",
        className = "webservice.CreateMaterial")
    @ResponseWrapper(localName = "createMaterialResponse", targetNamespace = "http://
        webservice/", className = "webservice.CreateMaterialResponse")
    public Material createMaterial(
        @WebParam(name = "material", targetNamespace = "")
        Material material)
        throws MaterialException_Exception
    ;
}
```

Výpis 3: Rozhranie príjemcu služby

Aby boli namespace jednotlivých služieb jedinečné, každa je umiestnená v package s jednoznačným názvom (Nájdete v moduloch package s názvom `webservice*`). Riešim tým problém s duplicitami, ktorý sa mohol objaviť i pri volaní v BPEL procese, ak by dva parametre rôznych webových služieb boli rovnako pomenované.

Orchestráciu som implementoval v jazyku **WSBPEL 2.0**. Daný orchester potom nasadím ako kompozitnú aplikáciu, ktorá beží v JBI kontajner pod Service Assemblies. Príklad BPEL procesu v zdrojovom kóde z obrázku 15 nájdete v prílohe B kód 4. Popis jednotlivých častí je v kapitole 5.1.1 Štruktúra definície procseu v BPEL.

Pre prezentačnú vrstvu som si vybral framework Struts 1.3.8. Implemetuje v sebe MVC model. Webové služby a kompozitné aplikácie volám v action metódach.

10.2 Štruktúra projektu

V prílohe A môžete vidiet' štruktúru modulu Sklad 34, pričom ostatné sú organizované rovnakým spôsobom. Balík `dao` obsahuje konfiguračnú triedu `Bean.xml` obsahujúcu údaje o bean dátach a prístupu do databáze, rozšírenie JDBCtemplaty a `InsertUpdater.java` pre tvorbu dotazov.

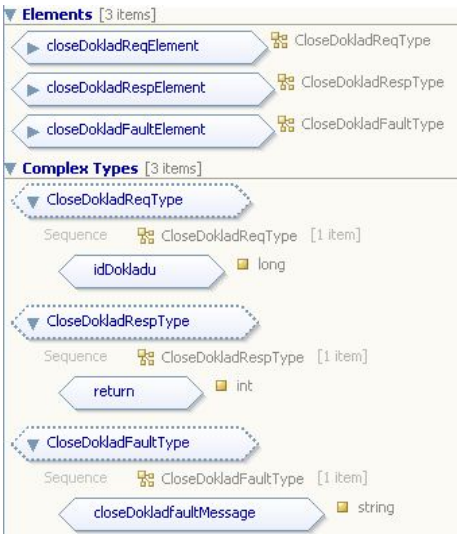
Pre každú tabuľku tu nájdeme DAO triedu a *Mapper* atribútov. V balík `model` obsahuje definície objektov, ktoré používa DAO vrstva a takisto metódy predstavujúce logiku webových služieb - `service`. Samostné webové služby sú umiestnené v záložke `webservice*`. `utils` zaobaluje zbierku pomocnej funkcionality a `exceptions` špeciálne výnimky jednolivých služieb.

Štruktúra BPEL projektov je veľmi jednoduchá. Obsahujú vygenerované partnerské wsdl, xsd súbory a vlastný popis, taktiež pomocou wsdl, xsd a bpel.

Projekt webovej aplikácie je rozdelený na časť s jsp stránkami a konfiguráciou (Web pages), odkazmi na klientov webových služieb (Web service reference) a samotný kód. V `Source package` nájdete triedy Äctionä "Form"(použitie v Struts frameworku) spolu s triedami modelu a pomocnými metódami.

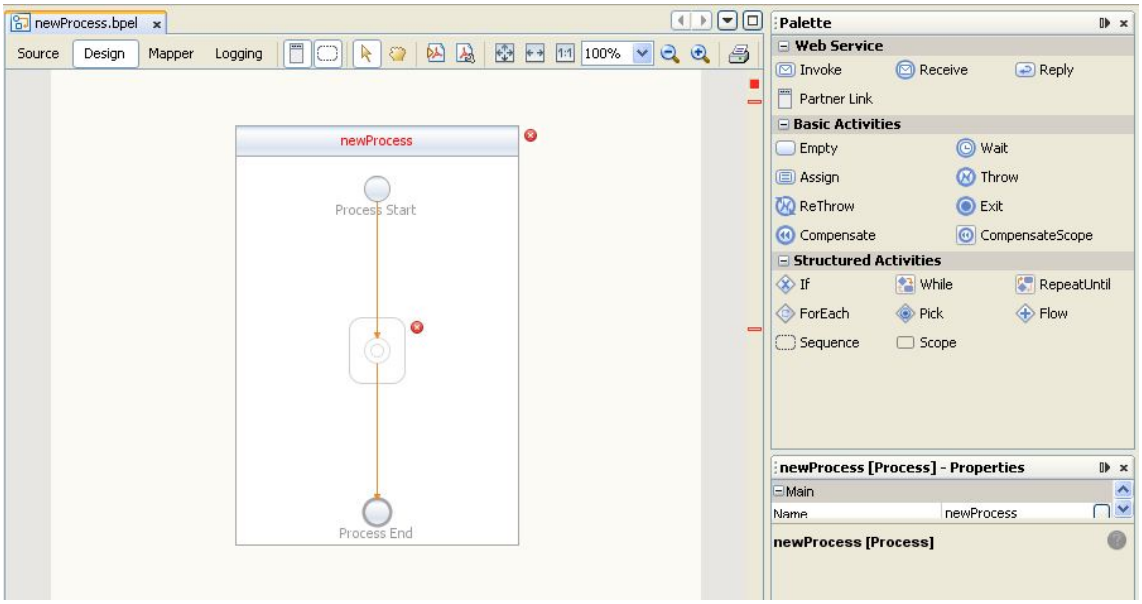
10.3 Príklad vytvorenia BPEL procesu

V NetBeanse otvoríme nový projekt z kategórie SOA: BPEL Module. Najskôr vytvoríme XML schéma, pomocou ktorého definujeme typy požiadavku, odpovede a chybovej hlášky. Vývojové prostredie obsahuje i vizuálny nástroj pre zjednodušenie práce. Do oblasti **Complex type** pridáme nové typy s elementami a definujeme ich. Complex type možu byt'i zanorené do zložitejších objektov. Do časti **Elements** vložíme elementy pre request, response a chybu a definujeme im typy z časti **Complex type**. Príklad xsd vidíte nižšie.



Obr. 18: XML Schema

Následne potrebujeme wsdl dokument, ktorým bude BPEL vystupovať voči okoliu. Pri jeho vytváraní importujeme XML Schemu z predchádzajúceho kroku a nadefinujeme časti Input, Output a Fault. Teraz nám ostáva už len samotný BPEL proces. Po vytvorení sa otvorí v BPEL Editore, záložka Design. Základ tvorí prázdny proces so sekvenciou, kam budeme pridávať jednotlivé aktivity z palety.



Obr. 19: Prázdny BPEL proces

Pre správne fungovanie musíme pridať:

1. partnerské linky - na ľavú stranu vložíme wsdl súbor BPEL procesu a vpravo umiestnime webové služby, ktoré budeme volať. Môže sa jednať i o iný BPEL proces (musi vystupovať ako webová služba).

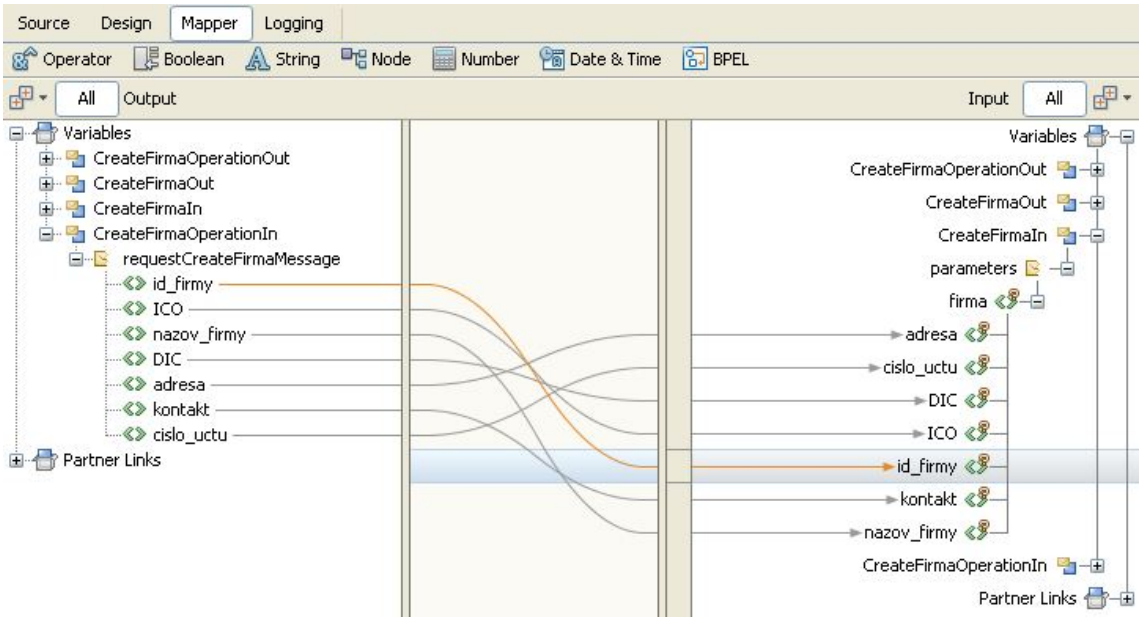
2. receive aktivitu - čakanie na príchodziu správu.

3. invoke - volanie metódy webovej služby.

4. reply - pre vrátenie odpovede z procesu.

Všetko je veľmi intuitívne a editor umožňuje automatické generovanie premenných a väzieb. Posledným krokom je previazanie jednotlivých aktivit pomocom `Assign`. K tomu použijeme záložku **Mapper** pre namapovanie správnych premenných, viď obrázok 20.

Aby sme mohli BPEL proces nasadiť a následne volať, potrebujeme ho zaobapridáme SOAP binding pre port a aplikáciu deployneme na server (príklad v prílohe A obrázok 35).



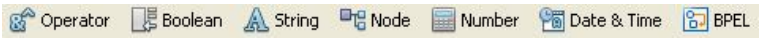
Obr. 20: Mapovanie premenných

10.4 Možnosti BPELu

Business Process Execution Language obsahuje prostriedky a riadiace bloky vhodné pre integráciu a koordináciu webových služieb. Nájdeme tu premenné, podmienky, cykly a vynimky. Takisto umožňuje prácu s reťazcami, číslami, dátumom, uzlami xml súborov a logické operácie.

10.4.1 Práca s premennými, Mapper editor

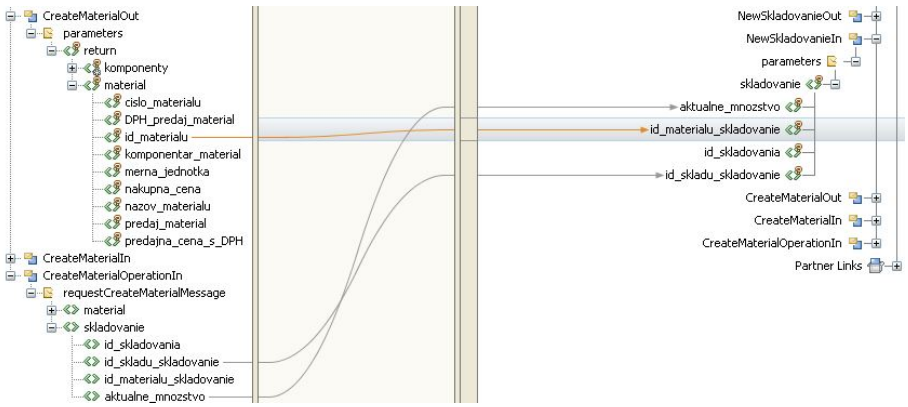
Premenné sú definované v časti `<variables>` pred samotným spustením procesu. Môžu byť XSD typu, alebo zodpovedajú vstupom a výstupom volaných partnerských linkov. Prácu a definovanie hodnôt umožňuje **Mapper** v BPEL editore pozostávajúci s nasledujúcich častí:



Obr. 21: Prvky v editore Mapper

Hodnoty priraďujeme (aktivita Assign) do premennej buď zo vstupu procesu, výstupu volanej služby (jedná sa tiež o premennú) alebo na základe aktivít a logických operácií. Tu môžeme definovať napríklad presný reťazec i konkrétne číslo. Ak sa nejedná o premennú primitívneho typu, môžeme ju naplniť vo viacerých krokoch a z rôznych zdrojov. Hodnoty sú totiž uchované celý beh procesu. Na obrázku 22 vidíte, že vstupom pre `newSkladovanie` sú dáta prijaté procesom a zároveň výstup metódy

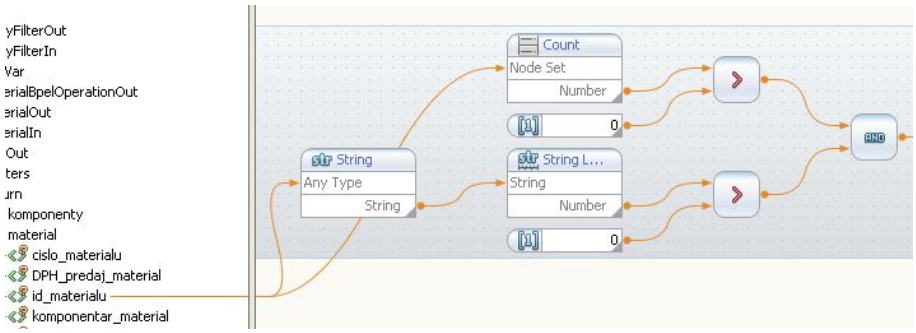
`createMaterial`. Id materiálu získame až po jeho vytvorení a následne ho previažeme so skladom.



Obr. 22: Mapovanie hodnôt z rôznych zdrojov

10.4.2 Príklady aktivít a operácií

BPEL podporuje väčinú známych aktivít a operácií známych s iných jazykov. Klasicky existujú logické operátory, práca s typmi boolean, string a number. Navyše umožňuje pracovať i z uzlami správ. Jeden rozdiel je, že nepozná **null** hodnotu. Musime preto skontrolavať, či daný uzol existuje a či obsahuje dáta.



Obr. 23: Kontrola existenicie uzla

V procese môžeme vykonávať aktivity paralelne vo viacerých tokoch. Na konci tohto flow čaká na všetky vetvy a pokračuje ďalej. Príklad vidíte v prílohe na obrázku 36, kde je znázornený `ziskBPEL`. Získame hodnoty nákupu, predaja a následne ich odčítame (v `Assign` aktivite). Ďalšími štandardnými konštrukciami sú cykly a podmienky, ktoré sa inicializáciou a prevedením neodlišujú od iných jazykov. Vstupy im definujeme pomocou Mapperu.

10.4.3 Výnimky

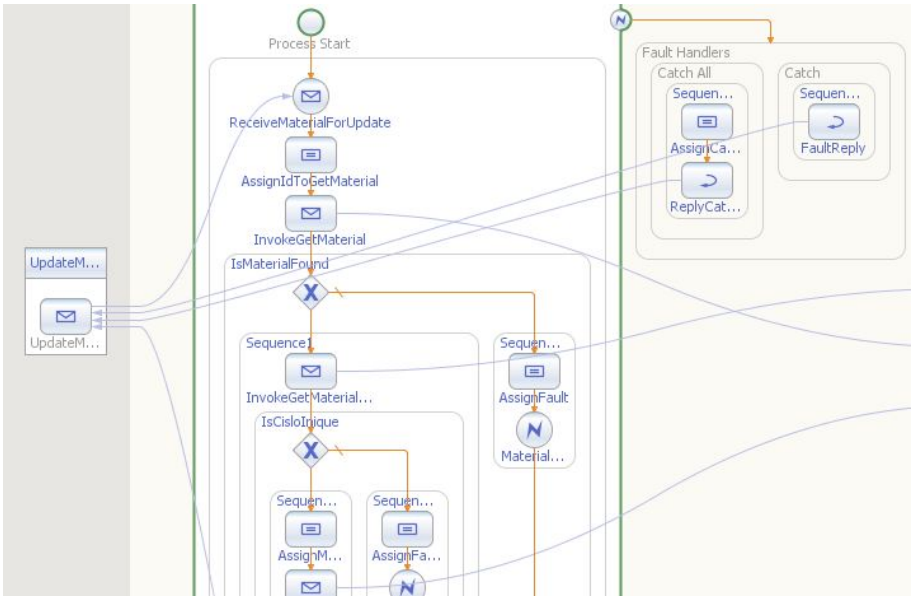
Komunikácia medzi webovými službami v rámci BPEL procesu väčšinou prebieha cez internet, ktorý je nie vždy spoľahlivý. Taktiež samotné služby môžu vyhodiť výnimku na základe logickej alebo runtime chyby. Z toho dôvodu musí BPEL podporovať správu týchto udalostí. Navyše proces signalizuje svoje vlastné typy chybových hlášok.

Zdroje výnimok v BPEL:

- `throw` aktivita - explicitné vyvolanie. Chceme vedome prerušiť beh neštandardným spôsobom.
- pri volaní operácie webovej služby - príde odpoveď v podobe WSDL chybovej správy, čo vyústi do BPEL výnimky. Príklad v `DeletePolozkaDokladuBPEL`.
- v rámci BPEL procesu - dôvodom môže byť konektivita, prostredie alebo logická chyba.

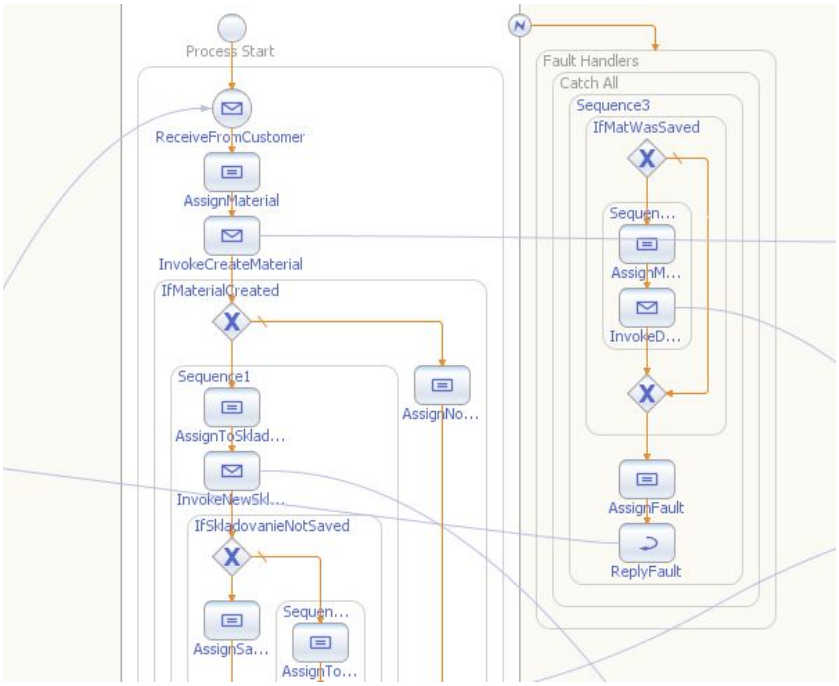
Zachytávanie

Na tento účel slúžia `fault handlery` definované pred spustením procesu. Používajú sa bloky `<catch>` pre odchytenie špecifickej výnimky (typ môže zodpovedať službe, ktorá ju vyvolá) a `<catchAll>`(len jeden pre celý scope). V nich definujeme aktivity, ktoré sa majú nasledne vykonať. Napríklad propagovanie hlášky klientovi (obrázok 24).



Obr. 24: Vyhodenie a zachytenie výnimky

Alebo zrušenie vykonaných zmien. Príklad v procese `<createMaterial>`. Ak nastane chyba, tak skontroluje, či už bol material uložený, ak áno odstráni ho (obrázok 25).



Obr. 25: Zachytenie výnimky a revertnutie zmien

V kóde odkiaľ BPEL voláme, môžeme zachytiť typ výnimky definovaný v XSD súbore.

10.5 Webová aplikácia

Pre vstup do systému je nutné zadať login a heslo. Na základe oprávení sa zobrazí menu s prístupnými položkami.

Obr. 26: Menu

V časti *Firmy* nájdete zoznam firiem a po kliknutí na *Nákupy* sa zobrazí ich prehľad. Tu môže užívateľ vytvárať nové nákupy materiálu. Systém automaticky aktualizuje množstvo materiálu na sklade, nákupnú cenu a sumu celého nákupu. Záložka *Sklady* obsahuje skladovanie materiálov, ich administráciu a tvorbu zložiek. Pre obsluhu slúži časť *Doklady*, kde pri vytváraní nových položiek vyberajú zo zoznamu predajných materiálov. Uzatvorením dokladu sa aktualizuje stav tovaru na sklade, v závislosti na jeho komponentách (najdôležitejší biznis proces). *Ekonomika* počíta obrat za zvolené obdobie.

Ukážky GUI nájdete v prílohe C.

10.6 Zhodnotenie

Business Process Execution Language je vhodné integračné riešenie z hľadiska nezávislosti na použitej platforme partnerských webových služieb. Jedná sa o štandard podporovaný mnohými veľkými IT korporáciami s rozsiahlov podporou. Orchestrácia pomôže sprehladniť procesy a s použitím správneho nástroja zjednoduší vývoj. Je na zvážení vývojárov, koľko logiky a validácie presunúť do procesu. Prílišná komplexnosť môže so sebou priniesť neprehľadnosť a náročnú udržovateľnosť.

Podrobnú dokumentáciu nájdete na stránkach firmy Oracle [13].

Pri diplomovej práci som použil nasledujúce nástroje:

- Netbeans 6.5.1
- Microsoft Visio 2010
- Oracle SQL Developer 2.1
- Oracle SQL Data Modeller 2.1

11 Záver

Výsledkom práce je zhrnutie problematiky orchestrácie a vytvorený prototyp informačného systému. Čítateľ by mal získať predstavu z oblasti webových služieb, servisne orientovanej architektúry a princípov orchestrácie. Nájde tu zoznam jazykov, platforiem a rámcov podporujúcich túto technológiu. Praktickú časť tvorí analýza a návrh systému, s implemetovanými vzorovými procesmi pomocou jazyka BPEL. Snažil sa ukázať jeho možnosti pri integrácii jednotlivých podsystémov. Výhoda mnou zvoleného riešenia spočíva v modelovaní procesov priamo spojenom s ich implementáciu. Navyše vývojové prostredie poskytuje interaktívny grafický editor, čo znamená jednoduchšiu konfigurovatelnosť, čitelnosť a orientáciu pri práci. Výsledok môže byť priamo predstavený a konzultovaný zo zákazníkom, bez nutnosti kompletnej implementácie vrátane GUI rozhrania, čo prináša finančnú úsporu a skráti dobu vývoja. Prezentačná vrstva je nezávislá od biznis logiky umiestnenej v BPEL procesoch a predstavuje rozhranie medzi systémom a užívateľom. Rozšírenie jednolivých modulov alebo pridanie ďalšej funkcionality nepredstavuje problém a záleží len na potrebách konkrétneho zákazníka.

Stanislav Žabka

12 Literatúra

[1] *Advanced Software Architecture Blog: Service Orchestration Guidelines* [online]. c2009, [cit.2011-07-28]. Dostupné z: http://leoshuster.blogspot.com/2009/01/service-orchestration-guidelines.html.

[2] Ing. Petr Weiss, Mgr. Marek Rychlý, *ARCHITEKTURA ORIENTOVANÁ NA SLUŽBY, NÁVRH ORIENTOVANÝ NA SLUŽBY, WEBOVÉ SLUŽBY* 1. vyd. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ, 2007.

[3] *Automa: Webové služby a ich implementácia v jazyku Java* [online]. c2010, [cit.2011-06-10]. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=34356.

[4] *Avio consulting: Oracle BPM and Oracle Service Bus Integration* [online]. c2010, [cit.2012-07-28]. Dostupné z: http://www.avioconsulting.com/blog/adesjardin/2010/02/24/oracle-bpm-and-oracle-service-bus-integration.

[5] *Bp: Building SOA work flows with BPM* [online]. c2011, [cit.2011-07-05]. Dostupné z: http://www.busmanagement.com/article/Building-SOA-work-flows-with-BPM/.

[6] *Enterprise Architecture: Workflow/Business Process Management (BPM) Service Pattern* [online]. c2007, [cit.2012-04-06]. Dostupné z: https://enterprisearchitecture.nih.gov/Pages/WorkflowServicePattern.aspx.

[7] *Fiorano: The ESB : Key Middleware Infrastructure for Business Components* [online]. c2012, [cit.2012-02-05]. Dostupné z: http://www.fiorano.com/products/esb-key-for-business-components.php.

[8] *Geoinformatics FCE CTU 2011* [online]. c2009, [cit.2011-06-22]. Dostupné z: http://geoinformatics.fsv.cvut.cz/gwiki/GUI_pro_orchestraci_GeoWebových_služeb.

[9] *IBM: WebSphere Enterprise Service Bus* [online]. c2012, [cit.2012-04-01]. Dostupné z: http://www-03.ibm.com/software/products/us/en/wsesb.

[10] *Masoud Kalali's Blog: Introducing OpenESB from development to administration and management.* [online]. c2012. Dostupné z: http://kalali.me/introducing-openesb-from-development-to-administration-and-management/.

[11] *MSDN: Orchestrating XML Web Services and Using the Microsoft .NET Framework with Microsoft BizTalk Server* [online]. c2002, [cit.2012-04-03]. Dostupné z: http://msdn.microsoft.com/en-us/library/ee251575(v=bts.10).aspx.

[12] *Oracle GlassFish Message Queue 4.4.2 Developer's Guide for Java Clients* [online]. c2010, [cit.2011-07-25]. Dostupné z: http://docs.oracle.com/cd/E19798-01/821-1796/aeqex/index.html.

[13] *Oracle Java CAPS BPEL Designer and Service Engine User's Guide* [online]. c2011. Dostupné z: http://docs.oracle.com/cd/E21454_01/html/821-2608/capsbpeldeseng_intro.html#scrolltoc.

[14] *Oracle Service-Oriented Architecture Suite* [online]. c2005, [cit.2013-02-25]. Dostupné z: http://www.arise-consulting.net/files/ORACLE%20SOA%20SUITE.pdf.

[15] *Oracle: Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI)* [online]. c2005, [cit.2013-02-25]. Dostupné z: http://www.oracle.com/technetwork/articles/javase/soa-142870.html.

[16] Jan Růžička, František Klímek, Michal Šeliga, Vladimír Maršík, Martin Prager, *Orchestrace geowebových služeb Metodika* 1. vyd. GAČR, 2009. 3 s. 205/07/0797.

[17] *Pieter Johan blog: Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI)* [online]. c2005, [cit.2012-02-25]. Dostupné z: http://pieterjohan.blogspot.sk/2008/02/service-oriented-architecture-soa-and.html.

[18] *Redhat: JBoss Enterprise SOA Platform* [online]. c2012, [cit.2012-01-15]. Dostupné z: http://www.redhat.com/resourcelibrary/whitepapers/jespvaluedif.

[19] *SWiK: SOA Process* [online]. c2009, [cit.2012-02-15]. Dostupné z: http://swik.net/SOA/SOA+Process.

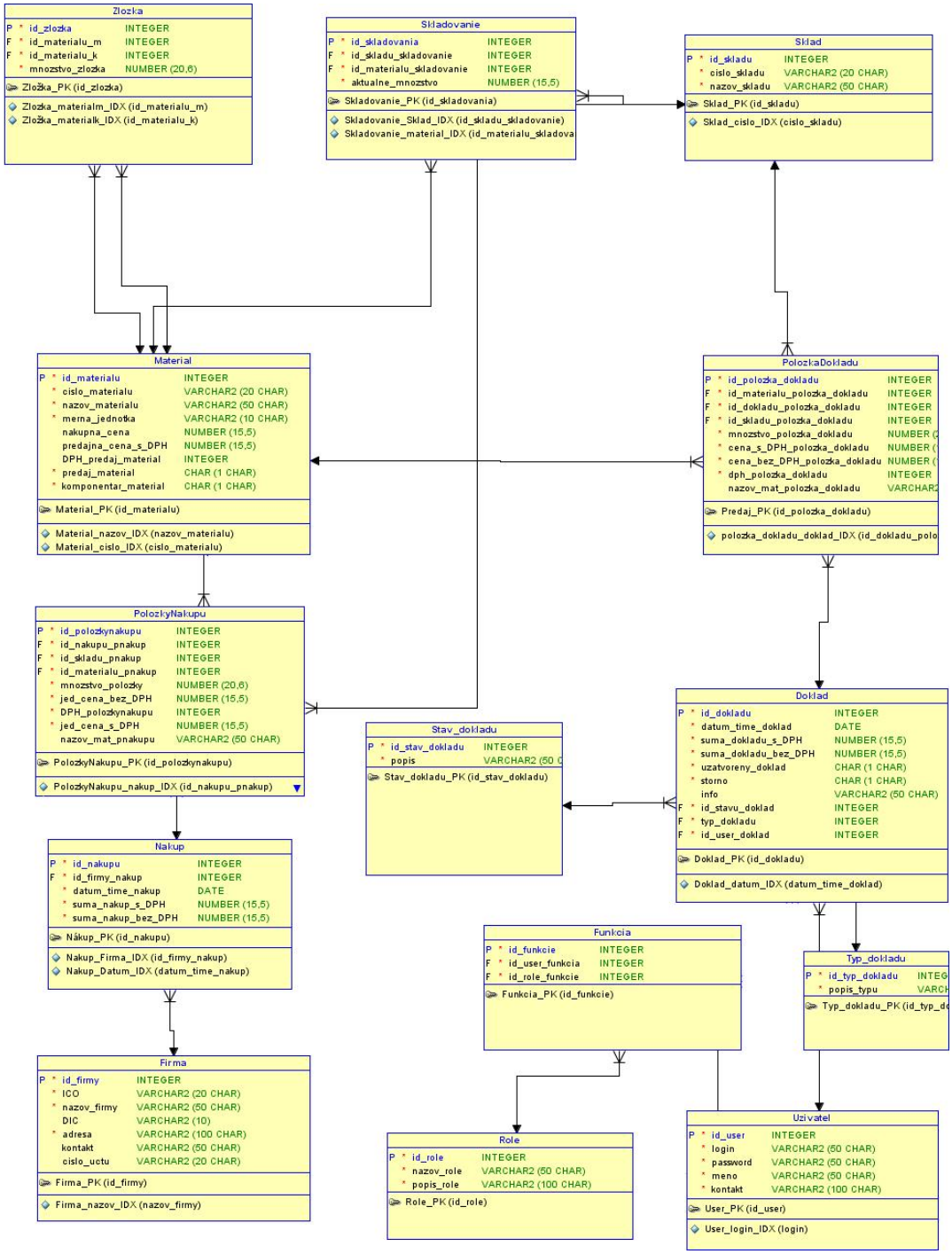
[20] Jeffrey Gortmaker, *The Advantages of Web Service Orchestration in Perspective* 1. vyd. ACM, 2004. 507 s. ISBN:1-58113-930-6.

[21] *W3: Web Services Description Language (WSDL) 1.1* [online]. c2001. Dostupné z: http://www.w3.org/TR/wsdl.

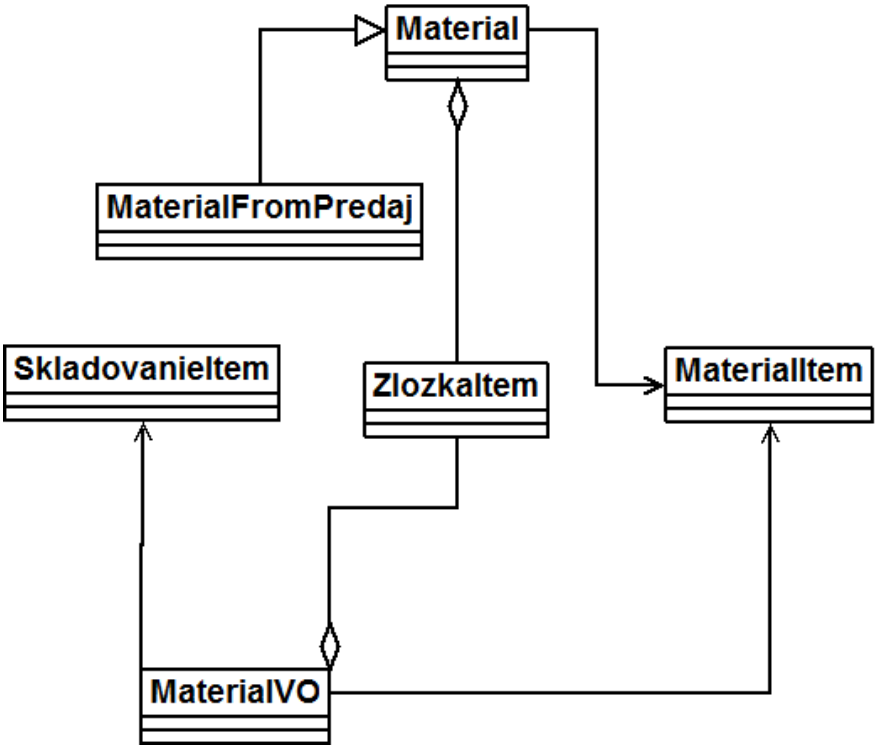
A Obrázky

	JBOSS ENTERPRISE SOA PLATFORM	IBM WEBSPHERE ESB	ORACLE SERVICE BUS
Podpora štandardu J2EE/Webové služby	x	x	x
Kompletná SOA platforma (ESB, workflow/orchestrácia, pravidlá, registre)	x	WebSphere Precess Server	Oracle SOA Suite
Open source	x		
Enterprise podpora	x	x	x
Biznis udalosťami-riadená architektúra	x	WebSphere Business Events	Oracle EDA Suite
Komplexné riadenie udalostí	BRMS CEP	WebSphere Business Events	Oracle EDA Suite CEP product
BPEL engine	BPEL tech	WebSphere Precess Server	Oracle SOA Suite
SOA	Registre +JON	prídavné produkty	prídavné produkty

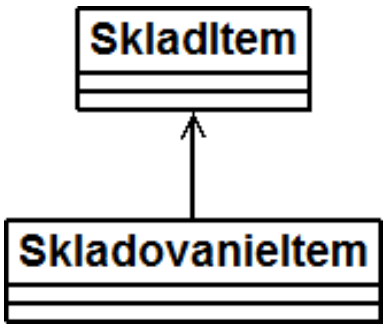
Obr. 27: Porovnanie SOA platforiem



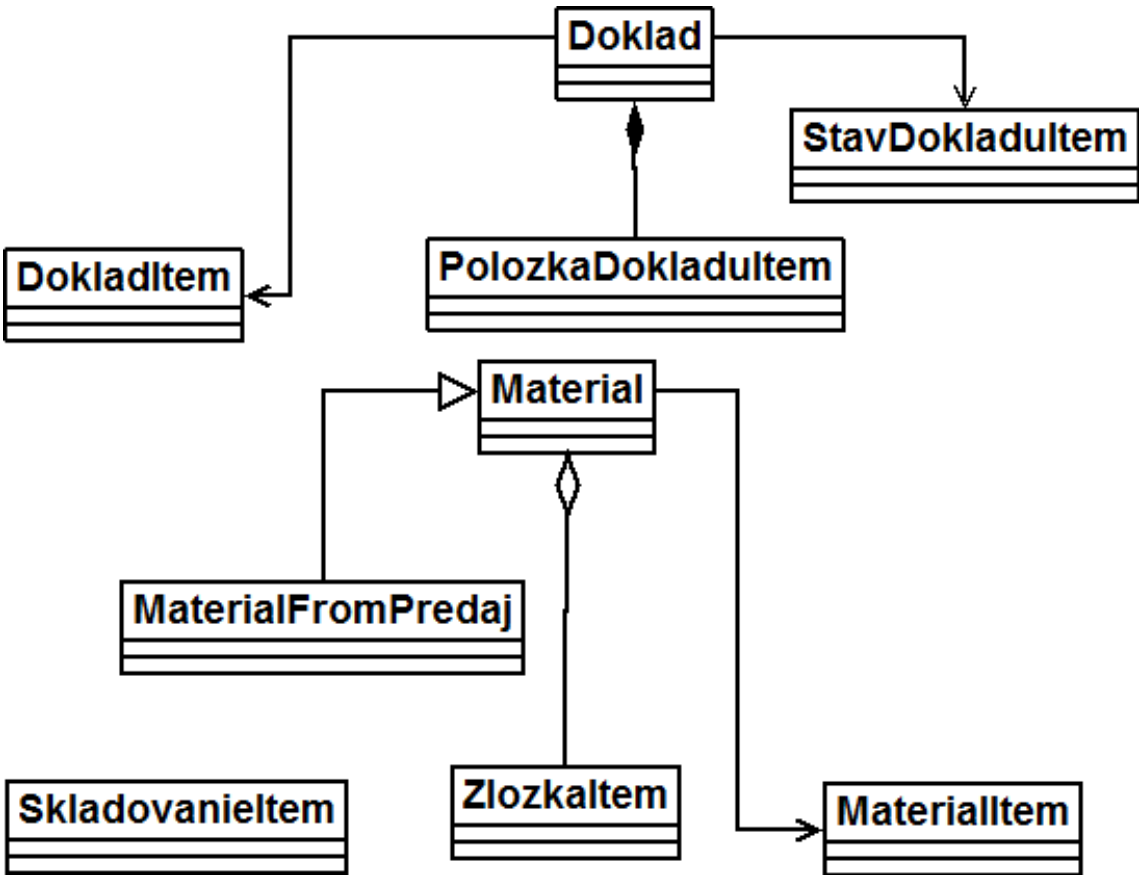
Obr. 28: Návrh databáze



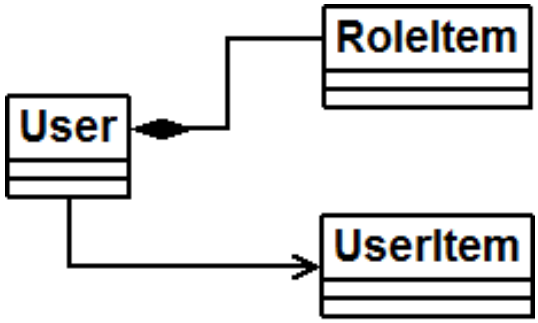
Obr. 29: Web modul materiál



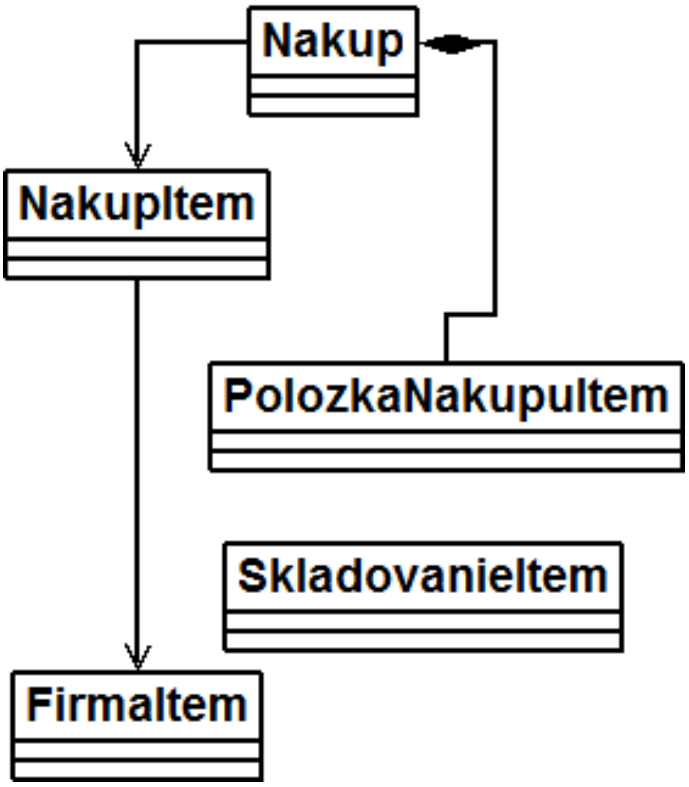
Obr. 30: Web modul sklad



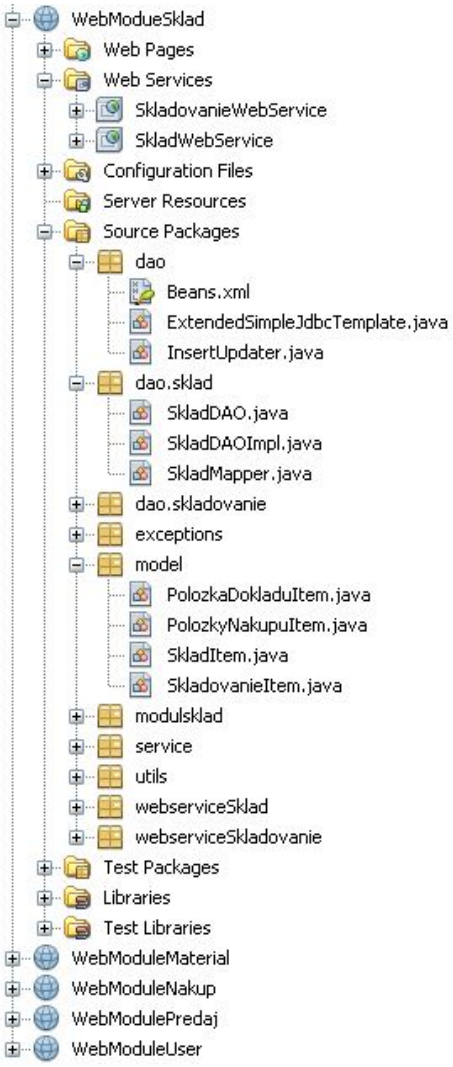
Obr. 31: Web modul predaj



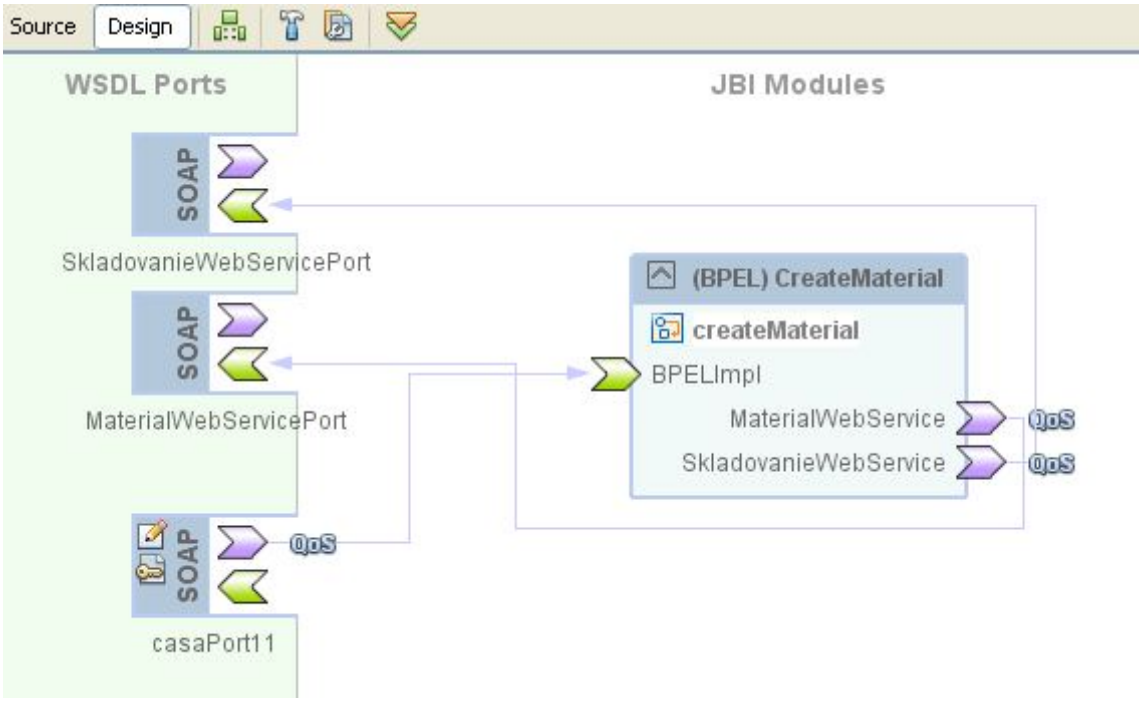
Obr. 32: Web modul užívateľ



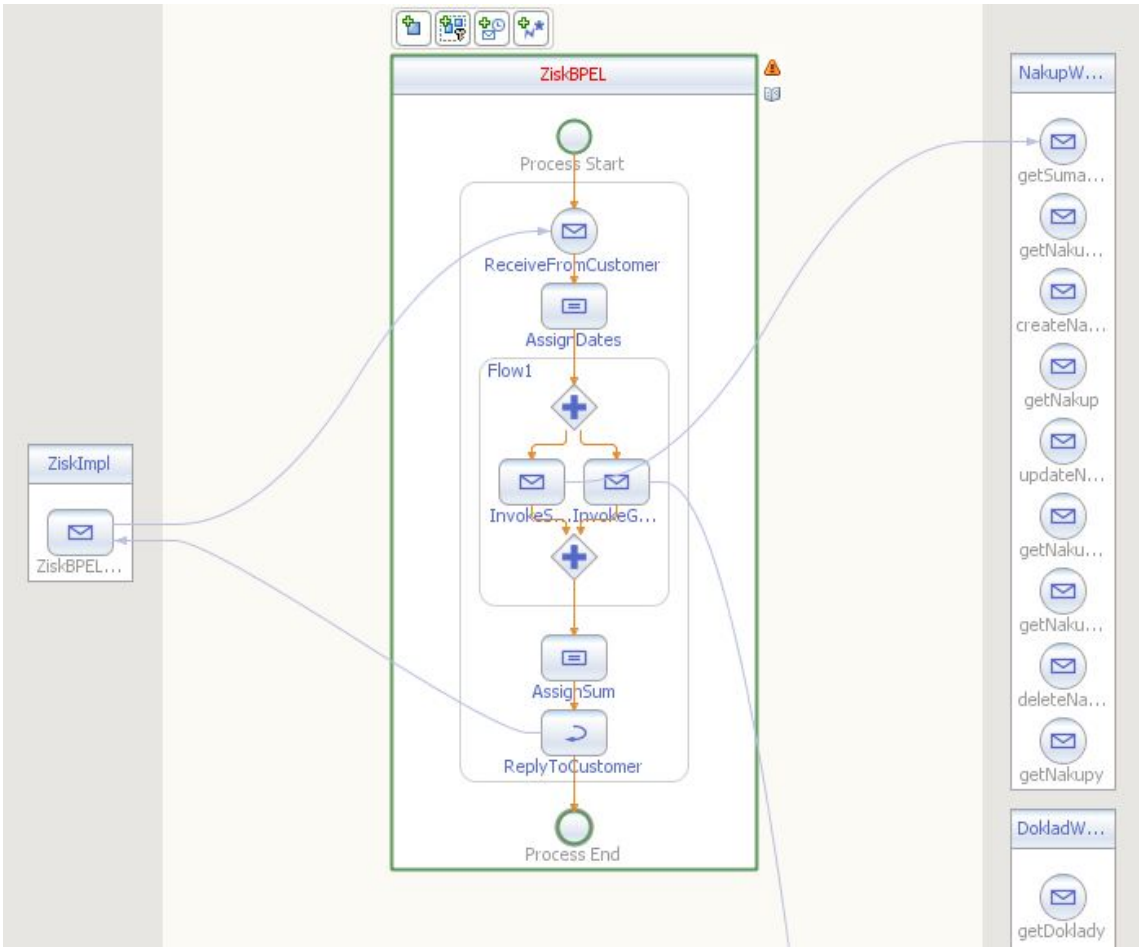
Obr. 33: Web modul nákup



Obr. 34: Štruktúra modulu



Obr. 35: Kompozitná aplikácia



Obr. 36: Volanie dvoch metód súčastne

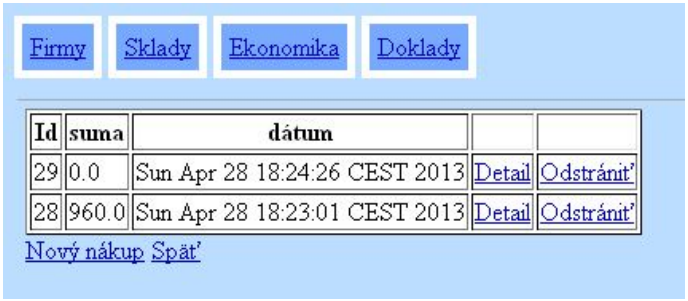
B Zdrojové kódy

```
<?xml version="1.0" encoding="UTF-8"?>
<process
    name="CreateFirma"
    targetNamespace="http://enterprise.netbeans.org/bpel/CreateFirma/CreateFirma"
    xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:sxt="http :// www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Trace"
    xmlns:sxed="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Editor"
    xmlns:tns="http :// enterprise.netbeans.org/bpel/CreateFirma/CreateFirma" xmlns:ns0="http://
        xml.netbeans.org/schema/CreateFirma">
    <import namespace="http://j2ee.netbeans.org/wsdl/CreateFirma" location="CreateFirma.wsdl"
        importType="http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="http://enterprise.netbeans.org/bpel/FirmaWebServiceServiceWrapper"
        location="Partners/FirmaWebService/FirmaWebServiceServiceWrapper.wsdl" importType=
        "http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="http://webservice/" location="Partners/FirmaWebService/
        FirmaWebServiceService.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
    <partnerLinks>
        <partnerLink name="FirmaWebService" xmlns:tns="http://enterprise.netbeans.org/bpel/
            FirmaWebServiceServiceWrapper" partnerLinkType="tns:FirmaWebServiceLinkType"
            partnerRole="FirmaWebServiceRole"/>
        <partnerLink name="ProcessImpl" xmlns:tns="http://j2ee.netbeans.org/wsdl/CreateFirma"
            partnerLinkType="tns:CreateFirma" myRole="CreateFirmaPortTypeRole"/>
    </partnerLinks>
    <variables>
        <variable name="CreateFirmaOperationOut" xmlns:tns="http://j2ee.netbeans.org/wsdl/
            CreateFirma" messageType="tns:CreateFirmaOperationResponse"/>
        <variable name="CreateFirmaOut" xmlns:tns="http://webservice/" messageType="tns:
            createFirmaResponse"/>
        <variable name="CreateFirmaIn" xmlns:tns="http://webservice/" messageType="tns:
            createFirma"/>
        <variable name="CreateFirmaOperationIn" xmlns:tns="http://j2ee.netbeans.org/wsdl/
            CreateFirma" messageType="tns:CreateFirmaOperationRequest"/>
    </variables>
    <sequence>
        <receive name="ReceiveFromCustomer" createInstance="yes" partnerLink="ProcessImpl"
            operation="CreateFirmaOperation" xmlns:tns="http://j2ee.netbeans.org/wsdl/
            CreateFirma" portType="tns:CreateFirmaPortType" variable="CreateFirmaOperationIn"
            />
        <assign name="CopyFromCustomer">
            <copy>
                <from>$CreateFirmaOperationIn.requestCreateFirmaMessage/ns0:id_firmy</from
                    >
                <to>$CreateFirmaIn.parameters/firma/id_firmy</to>
            </copy>
            <copy>
                <from>$CreateFirmaOperationIn.requestCreateFirmaMessage/ns0:ICO</from>
                <to>$CreateFirmaIn.parameters/firma/ICO</to>
            </copy>
            <copy>
```

```
                <from>$CreateFirmaOperationIn.requestCreateFirmaMessage/ns0:nazov_firmy</
                    from>
                <to>$CreateFirmaIn.parameters/firma/nazov_firmy</to>
            </copy>
            <copy>
                <from>$CreateFirmaOperationIn.requestCreateFirmaMessage/ns0:DIC</from>
                <to>$CreateFirmaIn.parameters/firma/DIC</to>
            </copy>
            <copy>
                <from>$CreateFirmaOperationIn.requestCreateFirmaMessage/ns0:adresa</from
                    >
                <to>$CreateFirmaIn.parameters/firma/adresa</to>
            </copy>
            <copy>
                <from>$CreateFirmaOperationIn.requestCreateFirmaMessage/ns0:kontakt</from
                    >
                <to>$CreateFirmaIn.parameters/firma/kontakt</to>
            </copy>
            <copy>
                <from>$CreateFirmaOperationIn.requestCreateFirmaMessage/ns0:cislo_uctu</
                    from>
                <to>$CreateFirmaIn.parameters/firma/cislo_uctu</to>
            </copy>
        </assign>
        <invoke name="InvokeCreateFirma" partnerLink="FirmaWebService" operation="
            createFirma" xmlns:tns="http://webservice/" portType="tns:FirmaWebService"
            inputVariable="CreateFirmaIn" outputVariable="CreateFirmaOut"/>
        <assign name="CopyResponseToCustomer">
            <copy>
                <from>$CreateFirmaOut.parameters/return/id_firmy</from>
                <to>$CreateFirmaOperationOut.responsePart/ns0:return</to>
            </copy>
        </assign>
        <reply name="ReplyToCustomer" partnerLink="ProcessImpl" operation="
            CreateFirmaOperation" xmlns:tns="http://j2ee.netbeans.org/wsdl/CreateFirma"
            portType="tns:CreateFirmaPortType" variable="CreateFirmaOperationOut"/>
    </sequence>
</process>
```

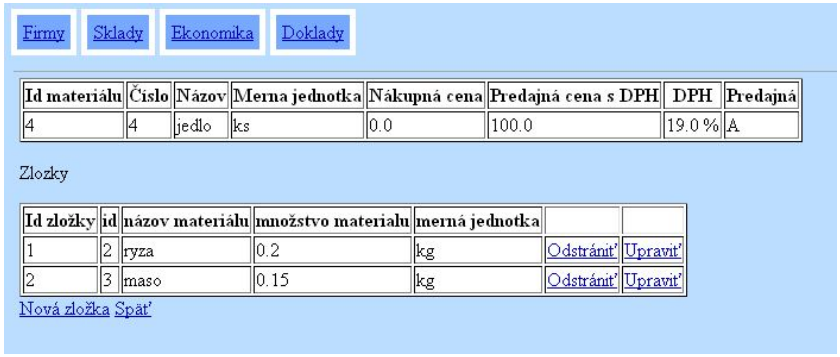
Výpis 4: Zdrojový kód BPEL procesu

C Webová aplikácia



Obr. 37: Firma a nákupy

Obr. 38: Nákup s položkami



Obr. 39: Materiál so zložkami

Firmy | Sklady | Ekonomika | Doklady

Id	Suma	Stav	Dátum		
1	120.0	2	Sun Mar 31 00:00:00 CET 2013	Detail	Odstránit'
2	400.0	5	Sat Apr 13 00:00:00 CEST 2013	Detail	Storno
3	120.0	2	Fri Apr 19 00:00:00 CEST 2013	Detail	Odstránit'
163	0.0	1	Tue Apr 23 00:00:00 CEST 2013	Detail	Odstránit'
181	600.0	1	Fri Apr 26 00:00:00 CEST 2013	Detail	Odstránit'
183	12000.0	1	Sat Apr 27 00:00:00 CEST 2013	Detail	Odstránit'
202	0.0	5	Sun Apr 28 00:00:00 CEST 2013	Detail	Storno
203	200.0	5	Sun Apr 28 00:00:00 CEST 2013	Detail	Storno
204	200.0	5	Sun Apr 28 00:00:00 CEST 2013	Detail	Storno
205	0.0	5	Sun Apr 28 00:00:00 CEST 2013	Detail	Storno
206	0.0	5	Sun Apr 28 00:00:00 CEST 2013	Detail	Storno
207	400.0	4	Sun Apr 28 00:00:00 CEST 2013	Detail	
208	200.0	4	Sun Apr 28 00:00:00 CEST 2013	Detail	
209	-200.0	5	Sun Apr 28 00:00:00 CEST 2013	Detail	Storno
210	-400.0	5	Sun Apr 28 00:00:00 CEST 2013	Detail	Storno
211	-200.0	5	Sun Apr 28 00:00:00 CEST 2013	Detail	Storno
212	0.0	4	Sun Apr 28 00:00:00 CEST 2013	Detail	
213	0.0	5	Sun Apr 28 00:00:00 CEST 2013	Detail	Storno
214	0.0	5	Sun Apr 28 00:00:00 CEST 2013	Detail	Storno
215	0.0	1	Sun Apr 28 00:00:00 CEST 2013	Detail	Odstránit'
216	500.0	5	Sun Apr 28 00:00:00 CEST 2013	Detail	Storno
217	300.0	5	Sun Apr 28 00:00:00 CEST 2013	Detail	Storno

Nový doklad

Obr. 40: Doklady

Doklady

Id	Dátum	Suma	Stav	Info
221	Mon Apr 29 00:00:00 CEST 2013	349.0	1	

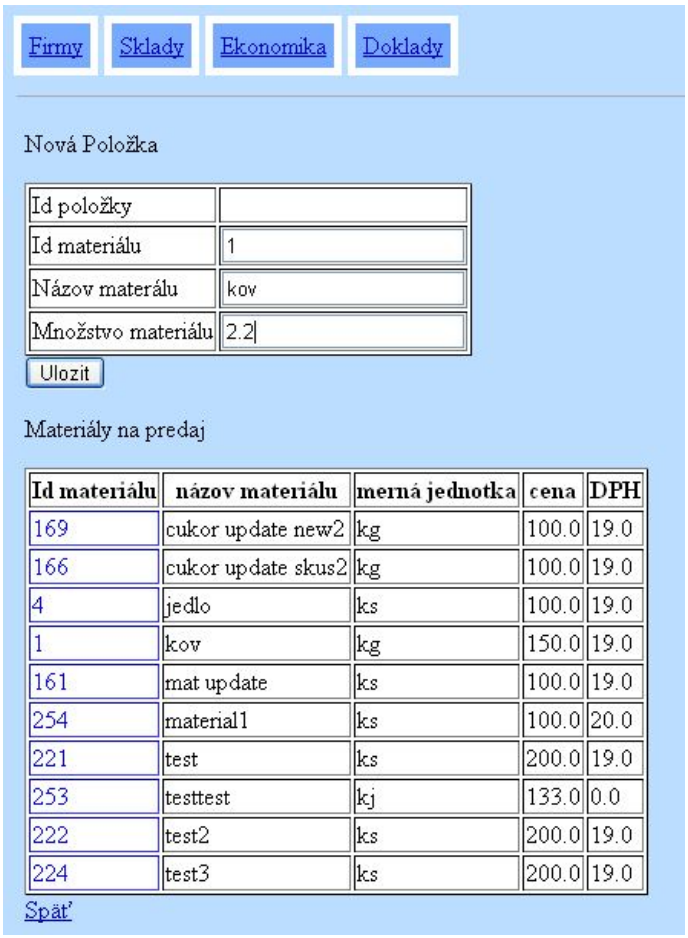
Uzavriet'

Položky

Id položky	id materiálu	Názov materiálu	Množstvo	Cena	
76	1000002	Pizza	3.0	315.0	Odstránit'
77	1000003	Vino	0.4	34.0	Odstránit'

Nová položka

Obr. 41: Doklad s položkami



Obr. 42: Nová položka